Facultad
de
Ciencias

LA ECUACIÓN DE PERONA-MALIK Y EL RUIDO EN LAS IMÁGENES DIGITALES
(The Perona-Malik equation and the noise in digital images)

Trabajo de Fin de Grado
para acceder al

GRADO EN MATEMÁTICAS

Autor: Jose Javier Ansorena Prieto

Director: Rafael Granero Belinchón

Septiembre - 2019

Resumen

Uno de los principales objetivos del procesamiento digital de imágenes es la reducción del ruido en estas. Esto tiene aplicaciones en varios campos, como la fotografía o la medicina. Sin embargo, el filtro más común para la eliminación de ruido, el filtro gaussiano, basado en la ecuación del calor, trae consigo una reducción de nitidez en toda la imagen, lo que puede originar una desaparición de detalles importantes de esta, o incluso la pérdida total de su significado.

En 1990, Pietro Perona y Jitendra Malik [10] propusieron un modelo de difusión, que llamaron anisotrópica, basado en la variante no lineal de la ecuación del calor, a partir del cual se puede construir un filtro que minimiza dicha pérdida de detalles mientras corrige el ruido de la imagen.

En esta memoria se estudiarán las propiedades matemáticas de la ecuación de Perona-Malik y se comparará con el filtro gaussiano. Además se aplicará a imágenes reales para probar su eficacia.

Palabras clave: filtro gaussiano, ecuación de Perona-Malik, bordes, imagen

Abstract

One of the main goals of digital image processing is the reduction of noise, which has applications in many fields, such as photography or medicine. However, the most common noise elimination filter, the Gaussian filter, based on the heat equation, involves a sharpness reduction in the entire image, which can lead to a disappearance of important details of it, or even the total loss of its meaning.

In 1990, Pietro Perona and Jitendra Malik [10] proposed a diffusion model, which they called anisotropic, based on the non-linear variant of the heat equation, from which a filter that minimizes said loss of detail while correcting the noise of the picture can be constructed.

In this essay, we will examine the mathematical properties of the Perona-Malik equation and compare it with the gaussian filter. It will also be applied to real images to prove its effectiveness.

Key words: Gaussian filter, Perona-Malik equation, edges, image

Índice

Notaciones

- Sea $n \in \mathbb{N}$, denotaremos por $v = (v_1, v_2, ..., v_n)^T$ al vector columna $n-$dimensional correspondiente. Es decir:

$$v := \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

  Para hablar de vectores fila, utilizaremos el vector traspuesto:

$$v^T := \begin{pmatrix} v_1 & v_2 & ... & v_n \end{pmatrix}$$

- Sea $f : \mathbb{R}^2 \to \mathbb{R}$:
  - Notación simplificada de derivadas parciales:

$$f_x(x, y) := \frac{\partial f(x, y)}{\partial x} \quad , \quad f_y(x, y) := \frac{\partial f(x, y)}{\partial y}$$

  - Gradiente $\nabla f(x, y) := (f_x(x, y), f_y(x, y))$.
  - Laplaciano $\Delta f(x, y) := f_{xx} + f_{yy}$.
  - Convolución:

    Dada otra función $g(x, y) : \mathbb{R}^2 \to \mathbb{R}$, se define la convolución de f y g como una nueva función $h(x, y)$:

$$h(x, y) = f(x, y) * g(x, y) := \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta) \cdot g(x - \alpha, y - \beta)\, d\alpha\, d\beta$$

- Sea $m \in \mathbb{N}$, y $k : \mathbb{R}^{m+1} \to \mathbb{R}$; $(x_1, x_2, ..., x_m, t) \to k(x_1, x_2, ..., x_m, t)$ donde entenderemos que las m primeras variables son espaciales, y la última temporal:

  Llamaremos

$$\nabla k := (k_{x_1}, k_{x_2}, ..., k_{x_m})$$

$$\big(\nabla k, k_t\big) := (k_{x_1}, k_{x_2}, ..., k_{x_m}, k_t)$$

$$\Delta k := k_{x_1 x_1} + k_{x_2 x_2} + ... + k_{x_m x_m}$$

- Operador divergencia:

  Dada una función $s : \mathbb{R}^2 \to \mathbb{R}^2;\ s(x,y) = \big(s_1(x,y), s_2(x,y)\big)$ se define la divergencia de s como

$$div(s(x,y)) := \frac{\partial s_1(x,y)}{\partial x} + \frac{\partial s_2(x,y)}{\partial y}$$

  Dada una función $q : \mathbb{R}^3 \to \mathbb{R}^2;\ q(x,y,t) = \big(q_1(x,y,t), q_2(x,y,t)\big)$, donde entenderemos que las dos primeras variables son espaciales, y la tercera temporal; se define la divergencia de q como

$$div(q(x,y,t)) := \frac{\partial q_1(x,y,t)}{\partial x} + \frac{\partial q_2(x,y,t)}{\partial y}$$

- Matriz Hessiana:

  Sea $n \in \mathbb{N}$, y $r : \mathbb{R}^{n+1} \to \mathbb{R}$; $(x_1, x_2, ..., x_n, t) \to r(x_1, x_2, ..., x_n, t)$, donde entenderemos que las n primeras variables son espaciales, y la última temporal. Se define la matriz Hessiana de r como:

$$\mathcal{H}r := \begin{pmatrix} \frac{\partial^2 r}{\partial x_1^2} & \frac{\partial^2 r}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 r}{\partial x_1 \partial x_n} & \frac{\partial^2 r}{\partial x_1 \partial t} \\ \frac{\partial^2 r}{\partial x_2 \partial x_1} & \frac{\partial^2 r}{\partial x_2^2} & \cdots & \frac{\partial^2 r}{\partial x_2 \partial x_n} & \frac{\partial^2 r}{\partial x_2 \partial t} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial^2 r}{\partial x_n \partial x_1} & \frac{\partial^2 r}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 r}{\partial x_n^2} & \frac{\partial^2 r}{\partial x_n \partial t} \\ \frac{\partial^2 r}{\partial t \partial x_1} & \frac{\partial^2 r}{\partial t \partial x_2} & \cdots & \frac{\partial^2 r}{\partial t \partial x_n} & \frac{\partial^2 r}{\partial t^2} \end{pmatrix}$$

- Producto escalar:

  Sea $n \in \mathbb{N}$, y sean $u = (u_1, u_2, ..., u_n)$ y $v = (v_1, v_2, ..., v_n)$ dos vectores $n-$dimensionales con coeficientes en $\mathbb{R}$; se define el producto escalar " $\cdot$ " de u y v como

$$u^T \cdot v = u_1 \cdot v_1 + u_2 \cdot v_2 + ... + u_n \cdot v_n$$

1. Introducción

En este trabajo se tratará de desarrollar métodos para el suavizamiento de "manchas" en imágenes en blanco y negro; es decir, la reducción en la cantidad de variaciones de intensidad entre píxeles vecinos.

Durante todo el trabajo, cada imagen se hará corresponder con una matriz bidimensional, I, y cada pixel de la imagen, con un elemento $I(x, y)$ de la matriz. Además, cada elemento de la matriz tomará un valor entero, entre 0 y 255, que indicará la claridad del pixel correspondiente, siendo 0 el negro, y 255 el blanco.

Por otro lado, muchas veces se hablará indistintamente de I como una imagen o como una matriz, y, también, como una función, que a cada (x, y) le asocia el valor $I(x, y)$.

A la hora de editar la imagen, se añadirá un nuevo parámetro t a la función I, para así irla modificando a medida que el nuevo parámetro varía. Es decir, se tratará de construir la función $I(x, y, t)$, donde $I(x, y, 0) = I_0(x, y)$ se corresponde con la imagen original.

1.1. Filtro Gaussiano y ecuación del calor

Uno de los procedimientos más comunes para construir esta función es el filtro gaussiano, que consiste en convolucionar, respecto a las variables espaciales, la imagen inicial con una función gaussiana

$$G(x, y, t) := \frac{1}{4c\pi t} \cdot e^{-\frac{x^2+y^2}{4ct}}$$

de varianza t, donde c es una constante real positiva, frecuentemente entre 0 y 1:

$$I(x, y, t) = I_0(x, y) * G(x, y, t) \tag{1}$$

A su vez, esta función puede ser vista como la solución de la ecuación del calor [12] [7]

$$I_t = div\,(c \cdot \nabla I) \tag{2}$$

con la condición inicial de que $I(x, y, 0)$ es la imagen conocida.

Como ya hemos dicho, el coeficiente c que multiplica a ΔI, llamado coeficiente de difusión, toma valor constante. Por ello se dice que el filtro gaussiano es lineal. Más adelante veremos como la idea básica de la ecuación de Perona-Malik es hacer que c pase a ser una función dependiente de las variables espaciales, y de la temporal.

La función (2) posee dos características que la hacen útil a la hora de cumplir nuestro objetivo:

1. **Causalidad:** El método se limita a corregir imperfecciones de la imagen cuando aumentamos el valor de t, y nunca generará nuevas manchas.

2. **Homogeneidad e Isotropía:** El suavizado se realizará de igual manera en todas las direcciones y no dependerá de la posición ni la orientación de la imagen.

Por tanto, con este método se puede conseguir suavizar la imagen y eliminar posibles manchas sobre ella. Sin embargo, este procedimiento presenta algunos inconvenientes, ya que no preserva eficientemente los bordes y se limita simplemente a darle homogeneidad al colorido de la imagen. Esto, tras aplicar el filtro un determinado número de veces (aumentar t una cierta cantidad), puede originar que se pierda completamente el significado de la imagen.

En muchas ocasiones, lo que se demanda es que la imagen originada preserve los "trazos" o "bordes", a la par que corrige los "puntos" o "manchas". Por ejemplo, en el ámbito de la medicina, las radiografías son imágenes relativamente complicadas de obtener, por lo que su calidad no siempre es la óptima. Si tratamos de aplicar el filtro gaussiano a una radiografía, es posible que, ya en bajos valores de t, las partes de dicha radiografía empiecen a mezclarse y se pierda la separación entre los distintos elementos de la imagen; mientras que a valores altos de t, la radiografía pierda todo su significado y se convierta en un borrón; por ejemplo, haciendo imposible localizar una fractura en medio de un área difuminada.

1.2. Aplicación del filtro a imágenes

A continuación vamos a exponer algunos ejemplos prácticos de imágenes procesadas mediante el filtro gaussiano, donde se aprecian los problemas que este puede generar:

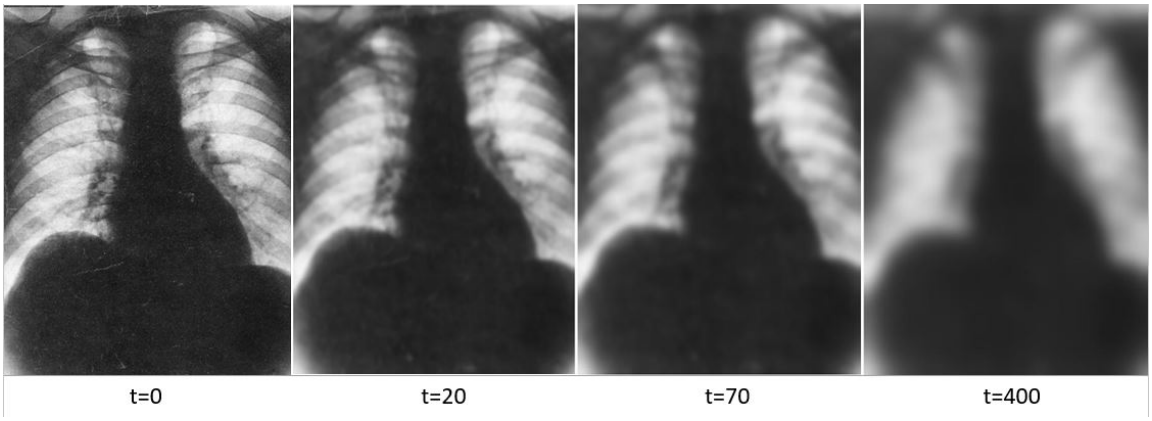

Figura 1: Imagen de un tórax con varias imperfecciones. Imagen original tomada de [5]

En la Figura 1 tenemos un ejemplo de una radiografia de un tórax con algunas imperfecciones, como en su parte central, la cual es negra, pero tiene algunas manchas blanquecinas, las cuales se corrigen al aplicar el filtro gaussiano (con $t = 70$ ya son prácticamente inapreciables), pero a la vez la imagen va perdiendo su significado.

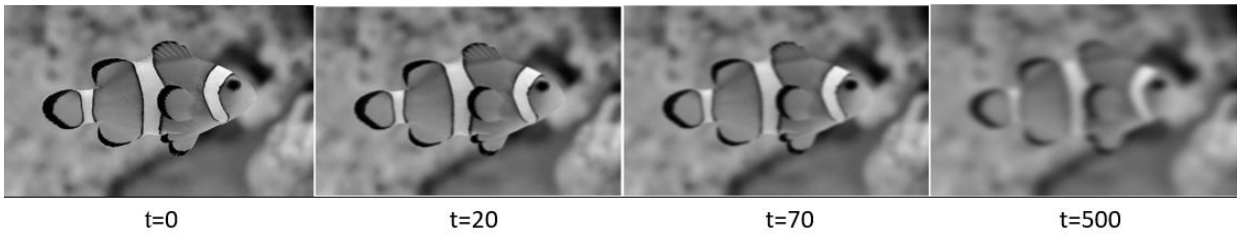

Figura 2: Imagen de un pez payaso. Imagen original tomada de [9].

En la Figura 2 se puede apreciar como las distintas trazas del pez se van difuminando las unas con las otras.

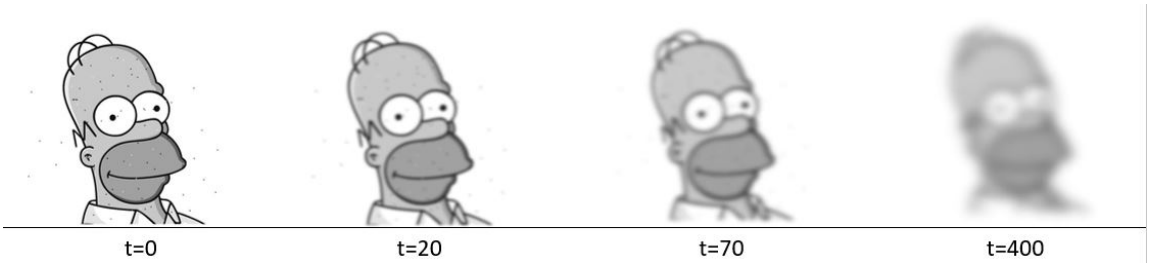

Figura 3: Imagen de Homer Simpson retocada con Paint para tratar de corregir las manchas. Imagen original tomada de [4]

La Figura 3 es una imagen manchada a propósito para intentar arreglarla mediante el filtro gaussiano. De nuevo, a medida que se corrigen las manchas, la imagen se va difuminando y los diferentes bordes y contornos se van perdiendo.

Para solucionar este problema, se propondrá un método similar al filtro gaussiano, pero que corrige este defecto mencionado.

2. Ecuación de Perona-Malik

Nuestro objetivo será construir un método que deberá satisfacer los siguientes principios [11]:

1. **Causalidad:** El método, al igual que el filtro gaussiano, deberá reducir la cantidad de variaciones de intensidad entre píxeles vecinos, y nunca aumentarla.
2. **Localización inmediata:** Para cada valor de t, los bordes deberán conservarse lo suficiente para que se preserve el "significado original de la imagen".
3. **Suavizado a trozos:** Para todo valor de t, el suavizado deberá hacerse separado por regiones; es decir, los bordes que separan distintas regiones de la imagen deberán preservarse más que los trazos dentro de una misma región.

El método que vamos a construir se basa, al igual que el filtro gaussiano, en la ecuación del calor, pero sin asumir que el coeficiente de difusión, c, sea constante. Es decir, será un modelo de difusión no lineal:

$$I_t = div\big(c(x,y,t)\cdot\nabla I\big) = c(x,y,t)\cdot\Delta I + \nabla c\cdot\nabla I \tag{3}$$

donde el operador de divergencia div, el gradiente ∇, y el Laplaciano Δ, se toman con respecto a las variables espaciales y no con respecto a t (ver apartado de Notaciones).

A la ecuación (3) la llamaremos ecuación de Perona-Malik, y será la que estudiaremos para obtener sus carácterísticas matemáticas y así poder construir un nuevo filtro que resulte efectivo.

Nótese que el caso particular en que $c(x,y,t)$ sea una función constante, la ecuación pasa a ser la del calor $I_t = c\cdot\Delta I$. Es decir, la ecuación de Perona-Malik es una generalización de la ecuación del calor.

Esta generalización hace que la ecuación sea ahora más complicada, pero a la vez, con una elección precisa de la función $c(x,y,t)$, seremos capaces de satisfacer en gran medida las condiciones de *Localización inmediata* y de *Suavizado a trozos* que le impusimos a nuestro método, sin perder la condición de *Causalidad* que cumplía inicialmente el filtro gaussiano. A continuación veremos cómo podemos construir esta función.

2.1. Suavizado a trozos y Localización inmediata

Para cumplir las condiciones de *Suavizado a trozos* y *Localización inmediata*, querríamos que la ecuación priorice el suavizado dentro de las regiones, y no en los bordes [10]. Esto podría conseguirse definiendo la función $c(x,y,t)$ de la siguiente manera:
c debería valer 1 en el interior de cada región, y 0 en los bordes; y su gradiente debería valer 0 tanto en el interior de las regiones, como en los bordes. De esta manera, tendríamos que en los bordes

$$I_t = 0\cdot\Delta I + (0,0)\cdot\nabla I = 0$$

es decir, la imagen no varía con respecto a t. A su vez, en el interior de las regiones

$$I_t = 1\cdot\Delta I + (0,0)\cdot\nabla I = \Delta I$$

es decir, se aplica el filtro gaussiano.

Sin embargo, como no conocemos los bordes de la imagen de antemano, la construcción de la función no será tan fácil ni tan exacta.

Para estimar la posición de los bordes, se construirá una función $E(x,y,t)$ definida sobre todas las imágenes, y tomando valores en $\mathbb{R}^2$; es decir, $E:\mathbb{R}^2\longrightarrow\mathbb{R}^2$, que, en principio, se debería definir como:

- $E(x,y,t) = (0,0)$ dentro de cada región.

- $E(x,y,t) = K(x,y,t) \cdot n(x,y,t)$ en cualquier punto del borde; donde $n(x,y,t)$ es un vector ortonormal al borde en el punto (x,y,t), y K es una función escalar. Por tanto, para poder construir dicho vector, tendremos que asumir que los bordes son curvas de clase C^1.
  $K(x,y,t)$ es la diferencia de intensidad entre las dos regiones que separa el borde. Es decir, sea (x_0,y_0,t_0) un punto del borde, A y B las regiones que separa dicho borde, y $S(A)$ y $S(B)$ la intensidad de un punto en la correspondiente región, suficientemente próximo a (x_0,y_0,t_0); se define

$$K(x_0,y_0,z_0) := |S(A) - S(B)|$$

Una vez se construya $E(x,y,t)$, el coeficiente de difusión se puede definir como $c(x,y,t) = g(\|E(x,y,t)\|)$; donde

$$g : [0,+\infty) \longrightarrow [0,1]$$

deberá ser una función derivable.

Además, como veremos a continuación, será necesario que $g(0) = 1$, que la función sea monótona decreciente y que tienda a 0.

Entonces, denotando $v(x,y,t) := ||E(x,y,t)|| :$

$$c_x = \frac{\partial g\left(v\left(x,y,t\right)\right)}{\partial x} = \frac{\partial g\left(v(x,y,t)\right)}{\partial\left(v(x,y,t)\right)} \cdot \frac{\partial v(x,y,t)}{\partial x} = g'(v(x,y,t)) \cdot \frac{\partial v(x,y,t)}{\partial x}$$

$$c_y = \frac{\partial g\left(v(x,y,t)\right)}{\partial y} = \frac{\partial g\left(v(x,y,t)\right)}{\partial\left(v(x,y,t)\right)} \cdot \frac{\partial v(x,y,t)}{\partial y} = g'\left(v(x,y,t)\right) \cdot \frac{\partial v(x,y,t)}{\partial y}$$

es decir:

$$\nabla c = g'\left(v(x,y,t)\right) \cdot \left(\frac{\partial v(x,y,t)}{\partial x}, \frac{\partial v(x,y,t)}{\partial y}\right)$$

De esta manera, tendremos que en el interior de cada región, $c(x,y,t) = g\left(v(x,y,t)\right) = g(0) = 1$. Además, elegiremos funciones g tales que $g'(0) = 0$, por lo que

$$\nabla c = 0 \cdot \left(\frac{\partial v(x,y,t)}{\partial x}, \frac{\partial v(x,y,t)}{\partial y}\right) = (0,0)$$

y, por tanto, $I_t = 1 \cdot \Delta I + (0,0) \cdot \nabla I = \Delta I$ y se realizará el filtro gaussiano.

Por otra parte, en los bordes tendremos que $c(x,y,t) = g\left(v(x,y,t)\right) = g\big(K(x,y,t)\big)$, que tenderá a 0 cuanto mayor sea $K(x,y,t)$, es decir, cuanto más marcado sea el borde (pudiendo valer 0 a partir de un cierto valor, dependiendo de como sea la función g). Igualmente, puesto que g es monótona y acotada, su derivada también tenderá a 0 (de nuevo, pudiendo valer 0 a partir de un cierto valor, dependiendo de como sea la función g). Por tanto, en bordes suficientemente marcados,

$$I_t \approx 0 \cdot \Delta I + (0,0) \cdot \nabla I = 0$$

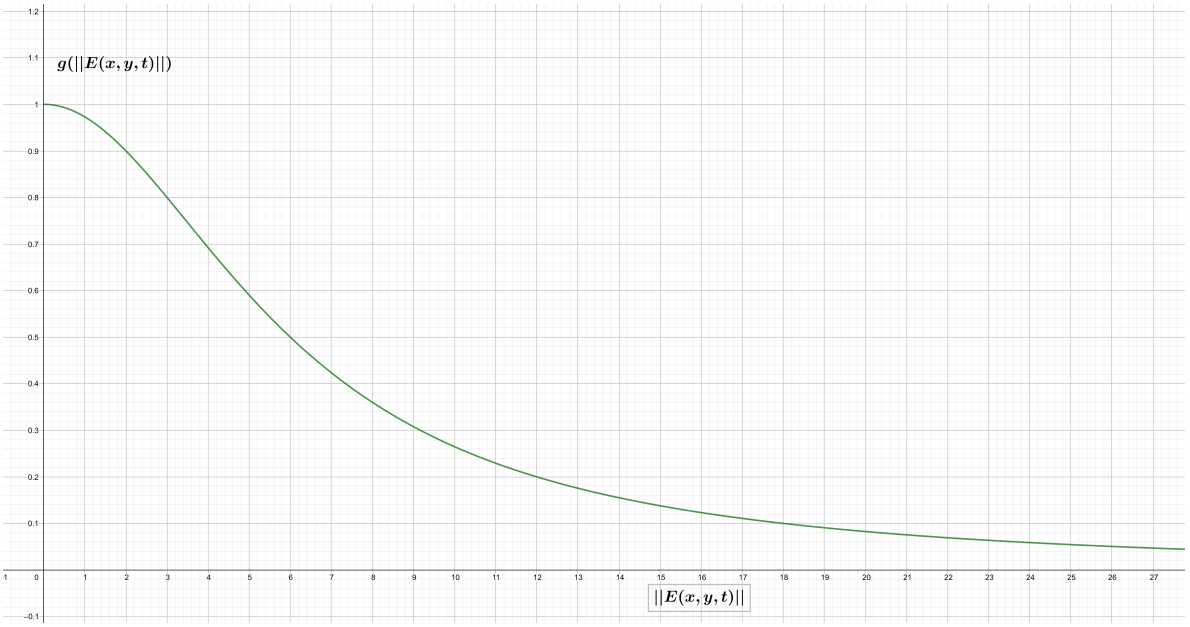

Figura 4: Ejemplo de función g, definida como

$$g\left(||E||\right) = \frac{1}{1 + \left(\frac{||E||}{6}\right)^2}$$

donde $g(0) = 1$, $g'(0) = 0$, y tanto g como g' tienden a 0 cuando $||E||$ tiende a infinito. Gráfica construida con GeoGebra Classic 6.

Como habíamos señalado, de esta manera se cumplirán la condiciones de *Suavizado a trozos* y *Localización inmediata*.

Como no conocemos la localización de los bordes de antemano, la función $E(x, y, t)$ deberá ser construida de forma aproximada.
Una primera idea sería elegir $E(x, y, t) = \nabla I(x, y, t)$, que, efectivamente:

- Dentro de cada región, no habrá cambios bruscos de intensidad, por lo que tanto I_x como I_y serán muy próximos a 0. Es decir, $\nabla I(x, y, t)$ será aproximadamente $(0, 0)$.
- En un punto (x_0, y_0, t_0) del borde, sabemos que

  $$(\nabla I(x_0, y_0, t_0), 1) = (I_x(x_0, y_0, t_0), I_y(x_0, y_0, t_0), 1)$$

  es ortogonal a la superficie $(x, y, I(x, y, t_0))$ en el punto (x_0, y_0, t_0), es decir, $\nabla I(x_0, y_0, t_0)$ es ortogonal al borde.
  Por otro lado,

  $$||\nabla I(x_0, y_0, t_0)|| = \sqrt{I_x^2(x_0, y_0, t_0) + I_y^2(x_0, y_0, t_0)}$$

  es una buena estimación del cambio de intensidad que se produce en el borde, ya que pondera de igual forma el cambio de intensidad en ambas direcciones.
  Por consiguiente, si descomponemos

  $$\nabla I(x_0, y_0, t_0) = ||\nabla I(x_0, y_0, t_0)|| \cdot \frac{\nabla I(x_0, y_0, t_0)}{||\nabla I(x_0, y_0, t_0)||}$$

tenemos que

$$\frac{\nabla I(x_0, y_0, t_0)}{||\nabla I(x_0, y_0, t_0)||}$$

es un vector unitario, y ortogonal al borde. Igualmente, $||\nabla I(x_0, y_0, t_0)||$ es una buena aproximación de la diferencia de intesidad entre las regiones $K(x_0, y_0, t_0)$. En conclusión, $\nabla I(x_0, y_0, t_0)$ será una buena aproximación de $E(x_0, y_0, t_0)$.

Por tanto, parece recomendable tomar $E(x, y, t) = \nabla I(x, y, t)$, y, como veremos más adelante, los resultados prácticos nos dan la razón.

2.2. Causalidad

Para verificar que la función cumple la condición de *Causalidad*, se probará un principio equivalente: el **Principio del Máximo**, que se obtiene como un corolario del teorema que veremos a continuación.

Sea A un abierto acotado de $\mathbb{R}^n$ y $T = (a, b)$ un intervalo de $\mathbb{R}$. Sea $D = A \times T = \{(x, t) : x \in A, t \in T\}$ el cilindro abierto de $\mathbb{R}^{n+1}$. Llamaremos ∂D al borde de D, $\overline{D}$ a su clausura, y ∂_TD, ∂_SD, ∂_BD a la parte superior, lateral e inferior de ∂D respectivamente:

$$\partial_T D = \{(x, t) : x \in A, t = b\}$$

$$\partial_S D = \{(x, t) : x \in \partial A, t \in (a, b]\}$$

$$\partial_B D = \left\{(x, t) : x \in \overline{A}, t = a\right\}$$

Además, llamaremos $\partial_{SD} D$ a la unión del borde lateral e inferior de D:

$$\partial_{SB} D = \partial_S D \cup \partial_B D$$

Entonces se tiene el siguiente teorema:

Teorema 1. *Dada una función $f : \mathbb{R}^{n+1} \to \mathbb{R}$ continua en $\overline{D}$, y dos veces diferenciable en $D \cup \partial_T D$; y dadas dos funciones $C, c : \mathbb{R}^{n+1} \to \mathbb{R}^+$ positivas, continuas en $\overline{D}$, y diferenciables en $D \cup \partial_T D$. Si f satisface la desigualdad:*

$$C(x, t) \cdot f_t - c(x, t) \cdot \Delta f - \nabla c \cdot \nabla f \leq 0 \tag{4}$$

en D, entonces el máximo de f en $\overline{D}$ se alcanza en $\partial_{SB} D$:

$$\max_{\overline{D}} f = \max_{\partial_{SB} D} f$$

Demostración:

▶ Primero consideraremos el caso en que f satisface la condición

$$C(x, t) \cdot f_t - c(x, t) \cdot \Delta f - \nabla c \cdot \nabla f < 0$$

Por hipótesis, f es continua en $\overline{D}$, que es un cerrado y acotado de $\mathbb{R}^{n+1}$, es decir, es compacto, y por tanto f tiene un máximo en él. Sea $p = (p_1, p_2, ..., p_n, t_0)$ ese máximo; por reducción al absurdo supondremos que $p \in D$, que es abierto, y por tanto p es un extremo relativo. Como f es dos veces diferenciable en D, tomando cualquier $v \in \mathbb{R}^{n+1}$ y $\varepsilon > 0$ suficientemente pequeño, podemos aproximar la función f en un punto mediante el polinomio de Taylor de grado 2:

$$f(p + \varepsilon v) = f(p) + \varepsilon \cdot \big(\nabla f(p), f_t(p)\big)^T \cdot v + \frac{\varepsilon^2}{2} \cdot v^T \cdot \mathcal{H}f(p) \cdot v + \mathrm{O}(\varepsilon^3) \leq f(p)$$

donde $\nabla f(p)$ es el gradiente de f en p con respecto a las n variables espaciales y $f_t(p)$ la derivada parcial respecto a la variable temporal. Además, $\mathcal{H}f(p)$ es la matriz Hessiana de f en p con respecto a las n variables espaciales, y a la temporal (ver apartado de Notaciones).

Como p es un extremo relativo, $\big(\nabla f(p), f_t(p)\big)$ es el vector nulo (en particular $f_t(p) = 0$), y, por tanto:

$$f(p) + \frac{\varepsilon^2}{2} \cdot v^T \cdot \mathcal{H}f(p) \cdot v + \mathrm{O}(\varepsilon^3) \leq f(p) \Rightarrow \frac{\varepsilon^2}{2} \cdot v^T \cdot \mathcal{H}f(p) \cdot v + \mathrm{O}(\varepsilon^3) \leq 0 \Rightarrow v^T \cdot \mathcal{H}f(p) \cdot v \leq 0$$

y como esto se cumple para cualquier $v \in \mathbb{R}^{n+1}$, la matriz $\mathcal{H}f(p)$ es semidefinida negativa. Por tanto, los elementos de su diagonal serán menores o iguales que 0. Particularmente, los n primeros elementos de su diagonal: $\frac{\partial^2 f(p)}{\partial x_1^2}$, $\frac{\partial^2 f(p)}{\partial x_2^2}$, ..., $\frac{\partial^2 f(p)}{\partial x_n^2}$, serán todos no positivos. Puesto que el Laplaciano de f es la suma de estos valores:

$$\Delta f(p) = \frac{\partial^2 f(p)}{\partial x_1^2} + \frac{\partial^2 f(p)}{\partial x_2^2} + ... + \frac{\partial^2 f(p)}{\partial x_n^2} \leq 0$$

Por tanto:

$$C(p) \cdot f_t(p) - c(p) \cdot \Delta f(p) - \nabla c(p) \cdot \nabla f(p) = -c(p) \cdot \Delta f(p) \geq 0$$

lo cual contradice la hipótesis.

De nuevo por reducción al absudo, suponemos que $p \in \partial_T D$, es decir, f alcanza el máximo en $t = b$. En este caso, demostraremos que $f_t(p) \geq 0$. Para ello, construiremos una sequencia de puntos $\tau_i = (p_1, ..., p_n, \dot{\tau}_i)$, con $\dot{\tau}_i \in (a, b)$, que converja a p, y con $f_t(\tau_i) \geq 0 \ \ \forall i \in \mathbb{N}$.

Definimos $\psi_1 = (p_1, ..., p_n, \dot{\psi}_1)$, con $\dot{\psi}_1 = \frac{b}{2} + \frac{a}{2}$.
Por el Teorema del Valor Medio para $n + 1$ variables, sabemos que existe un punto, $\tau_1 = (p_1, ..., p_n, \dot{\tau}_1)$, con $\dot{\tau}_1 \in (\dot{\psi}_1, b)$, tal que $f(p) - f(\psi_1) = \big(\nabla f(\tau_1), f_t(\tau_1)\big)^T \cdot (p - \psi_1)$.

Como p es máximo, se tiene que

$$\big(\nabla f(\tau_1), f_t(\tau_1)\big)^T \cdot (p - \psi_1) = \big(f_{x_1}(\tau_1), ..., f_{x_n}(\tau_1), f_t(\tau_1)\big) \cdot \left(0, 0, ..., 0, \frac{b}{2} - \frac{a}{2}\right) = f_t(\tau_1) \cdot \left(\frac{b}{2} - \frac{a}{2}\right) \geq 0$$

Es decir,

$$f_t(\tau_1) \geq 0$$

Ahora, para cualquier $i \in \mathbb{N}$ definimos $\psi_i = (p_1, ..., p_n, \dot{\psi}_i)$, con $\dot{\psi}_i = \frac{(2^i-1)\cdot b}{2^i} + \frac{a}{2^i}$

Aplicando de nuevo el Teorema del Valor Medio, obtenemos un $\tau_i = (p_1, ..., p_n, \dot{\tau}_i)$, con $\dot{\tau}_i \in (\dot{\psi}_i, b)$, tal que $f(p) - f(\psi_i) = \big(\nabla f(\tau_i), f_t(\tau_i)\big)^T \cdot (p - \psi_i)$.

Por tanto, se tiene que

$$(\nabla f(\tau_i), f_t(\tau_i))^T \cdot (p - \psi_i) = f_t(\tau_i) \cdot \left(b - \left(\frac{(2^i-1)\cdot b}{2^i} + \frac{a}{2^i}\right)\right) = f_t(\tau_i) \cdot \left(\frac{b}{2^i} - \frac{a}{2^i}\right) \geq 0$$

Es decir,

$$f_t(\tau_i) \geq 0$$

Finalmente, cuando i tiende a infinito, sabemos que $\frac{(2^i-1)\cdot b}{2^i}$ tiende a b, es decir, $\dot{\psi}_i$ tiende a b, y, por tanto, también $\dot{\tau}_i$. Entonces, τ_i es una sucesión de puntos que converge a p, y $f_t(\tau_i) \geq 0 \ \ \forall i \in \mathbb{N}$, con f_t una función continua, por lo que $f_t(p) \geq 0$.

Además se sigue cumpliendo que $\nabla f(p)$ es el vector nulo y $\Delta f(p) \leq 0$.
Entonces:

$$C(p) \cdot f_t(p) - c(p) \cdot \Delta f(p) - \nabla c(p) \cdot \nabla f(p) = C(p) \cdot f_t(p) - c(p) \cdot \Delta f(p) \geq 0$$

y de nuevo se llega a un absurdo, concluyendo así este caso.

► Para el caso más general $C(x,t) \cdot f_t - c(x,t) \cdot \Delta f - \nabla c \cdot \nabla f \leq 0$, definimos las funciones $g_m(x,t) = f(x,t) - \dfrac{1}{m}(t-a)$ para cualquier $m \in \mathbb{N}$.

Tenemos que $(g_m)_t = f_t - \dfrac{1}{m}$, $\quad \nabla g_m = \nabla f \quad$ y $\quad \Delta g_m = \Delta f$, y por tanto:

$$\begin{aligned} &C(x,t) \cdot (g_m)_t - c(x,t) \cdot \Delta g_m - \nabla c \cdot \nabla g_m = \\ &= C(x,t) \cdot f_t + \frac{1}{m} \cdot C(x,t) - c(x,t) \cdot \Delta f - \nabla c \cdot \nabla f < \\ &< C(x,t) \cdot f_t - c(x,t) \cdot \Delta f - \nabla c \cdot \nabla f \leq 0 \end{aligned}$$

es decir, g_m cumple el caso anterior, por lo que:

$$\max_{\overline{D}} g_m = \max_{\partial_{SB} D} g_m \quad \forall m \in \mathbb{N}$$

Además, $f = g_m + \dfrac{1}{m}(t-a) \leq g_m + \dfrac{1}{m}(b-a)$, y $f = g_m + \dfrac{1}{m}(t-a) \geq g_m$ en $\overline{D}$, y entonces:

$$\max_{\overline{D}} f \leq \max_{\overline{D}} \left(g_m + \frac{1}{m}(b-a)\right) = \max_{\overline{D}}(g_m) + \frac{1}{m}(b-a) = \max_{\partial_{SB} D}(g_m) + \frac{1}{m}(b-a) \leq \max_{\partial_{SB} D}(f) + \frac{1}{m}(b-a)$$

Finalmente, haciendo m tender a ∞ obtenemos:

$$\max_{\overline{D}} f = \lim_{m \to \infty} \left(\max_{\overline{D}} f\right) \leq \lim_{m \to \infty} \left(\max_{\partial_{SB} D}(f) + \frac{1}{m}(b-a)\right) = \max_{\partial_{SB} D} f$$

que es el resultado buscado. □

Dadas todas las notaciones del **Teorema 1**, para el siguiente corolario supondremos que A es un abierto acotado de $\mathbb{R}^n$ de la forma $A = (a_1, b_1) \times (a_2, b_2) \times ... \times (a_n, b_n)$; es decir, un "rectángulo" $n-$dimensional de lados paralelos a los ejes.
Además, definiremos una nueva superficie, $\overset{\sim}{\partial_S} D$, como:

$$\overset{\sim}{\partial_S} D = \left\{ (x_1, x_2, ..., x_n, t) \in \partial_S D \text{ tales que } t \in (a, b) \text{ y } \exists!\ i \in \{1, 2, ..., n\} : x_i \in \{a_i, b_i\} \right\}$$

$\overset{\sim}{\partial_S} D$ es la parte lateral de ∂D a la que se le han quitado los puntos donde, en principio, no tiene sentido hablar de vector normal. Es decir, se eliminan las "esquinas" para que la superficie sea suficientemente suave.

Por tanto, para todo

$$(x_1, x_2, ..., x_n, t) \in \overset{\sim}{\partial_S} D$$

ahora podemos definir $u(x_1, x_2, ..., x_n, t)$ como el vector normal unitario exterior a $\overset{\sim}{\partial_S} D$ en dicho punto.

Con todo esto, ya podemos enunciar y demostrar el Principio del Máximo, como un corolario del teorema anterior:

Corolario 1 (**Principio del Máximo**). *Dadas todas las notaciones anteriores. Suponiendo que f satisface las hipótesis del **Teorema 1**, y que es dos veces diferenciable en $\partial_S D$. Si f satisface las condiciones de contorno de Neumann sobre $\overset{\sim}{\partial_S} D$; es decir, si*

$$(\nabla f, f_t)(x_1, x_2, ..., x_n, t) \cdot u(x_1, x_2, ..., x_n, t) = 0 \quad \forall (x_1, x_2, ..., x_n, t) \in \overset{\sim}{\partial_S} D$$

entonces se tiene que

$$\max_{\overline{D}} f = \max_{\partial_B D} f$$

Demostración:

Como f satisface el **Teorema 1**, tenemos que

$$\max_{\overline{D}} f = \max_{\partial_{SB} D} f$$

▶ Al igual que en **Teorema 1**, primero consideraremos el caso en que f satisface la condición

$$C(x, t) \cdot f_t - c(x, t) \cdot \Delta f - \nabla c \cdot \nabla f < 0$$

Por reducción al absurdo, supondremos que $\max_{\overline{D}} f = p = (p_1, p_2, ..., p_n, t_0) \in \partial_S D$.

A continuación, dado cualquier punto $q = (q_1, q_2, ..., q_n, t_1) \in \partial_S D$ tal que $t_1 \in (a, b)$, demostraremos que $f_{x_i}(q) = 0 \quad \forall i :\ q_i \in \{a_i, b_i\}$. La demostración se hará de una forma recursiva sobre el número de elementos, $k \in \{1, 2, ..., n\}$, tales que $q_i \in \{a_i, b_i\}$:

- Si $q \in \widetilde{\partial_S D}$, tenemos que $t_1 \in (a,b)$ y $k=1$, es decir, $\exists!\, i \in \{1,2,...,n\}$ tal que $q_i \in \{a_i, b_i\}$. Fijado i, supongamos, por ejemplo, que $q_i = b_i$ (el caso $q_i = a_i$ es análogo). Entonces $q = (q_1, q_2, ..., q_{i-1}, b_i, q_{i+1}, ..., q_n, t_1)$.
  Como el rectángulo $(n+1)-$dimensional tiene los lados paralelos a los ejes de coordenadas, el vector normal en cualquier punto de la superficie

$$\{(x_1, x_2, ..., x_n, t) \in \widetilde{\partial_S D} \text{ tales que } x_i = b_i\}$$

  será el $u(x_1, x_2, ..., x_n, t) = (0, 0, ..., 0, \underset{i)}{1}, 0, ..., 0)$. En particular $u(q) = (0, 0, ..., 0, \underset{i)}{1}, 0, ..., 0)$.

  Por tanto, como f satisface las condiciones de contorno de Neumann sobre $\widetilde{\partial_S D}$, se tiene que

$$(\nabla f, f_t)(q) \cdot u(q) = 0$$

  es decir

$$f_{x_1}(q)\cdot 0 + f_{x_2}(q)\cdot 0 + ... + f_{x_{i-1}}(q)\cdot 0 + f_{x_i}(q)\cdot 1 + f_{x_{i+1}}(q)\cdot 0 + ... + f_{x_n}(q)\cdot 0 + f_t(q)\cdot 0 = f_{x_i}(q) = 0$$

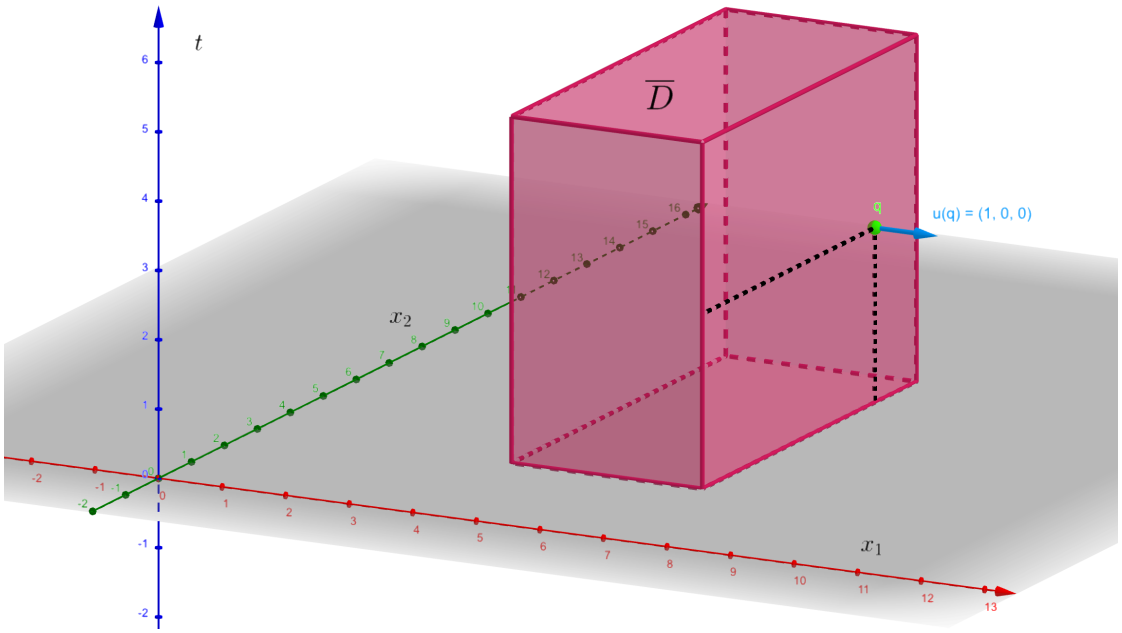

Figura 5: Ejemplo gráfico de una posible $\overline{D}$, con $n=2$, y un punto q del borde lateral, donde el vector normal es el $(1,0,0)$. Figura construida con GeoGebra Classic 6.

- En el caso recursivo, suponiendo que se cumple para $k = m-1$, lo probaremos para $k = m$.
  Si $k = m$, con $m \in \{2, ..., n\}$, existen exactamente m elementos distintos $i_1, ..., i_m \in \{1, 2, ..., n\}$ tales que $q_{i_1} \in \{a_{i_1}, b_{i_1}\}, ..., q_{i_m} \in \{a_{i_m}, b_{i_m}\}$; supondremos, sin pérdida de generalidad, que $q_{i_1} = b_{i_1}, ..., q_{i_m} = b_{i_m}$, es decir, $q = (q_1, ..., b_{i_1}, ..., b_{i_2}, ..., b_{i_m}, ..., q_n, t_1)$.

Dado cualquier punto

$$\check{q} = (q_1, ..., q_{i_1-1}, \check{q_{i_1}}, q_{i_1+1}, ..., b_{i_2}, ..., b_{i_m}, ..., q_n, t_1) \text{ con } \check{q_{i_1}} \in \left(\frac{a_{i_1} + b_{i_1}}{2}, b_{i_1}\right)$$

tenemos que $\check{q}$ satisface las condiciones del caso $k = m - 1$, y, por tanto

$$f_{x_{i_\theta}}(\check{q}) = 0 \ \ \forall \theta \in \{2, 3, ..., m\}$$

Por hipótesis, f es dos veces diferenciable; es decir, $f_{x_{i_\theta}}$ es una función continua. Además,

$$f_{x_{i_\theta}}(q_1, ..., q_{i_1-1}, \check{q_{i_1}}, q_{i_1+1}, ..., b_{i_2}, ..., b_{i_m}, ..., q_n, t_1) = 0 \ \ \forall \check{q_{i_1}} \in \left(\frac{a_{i_1} + b_{i_1}}{2}, b_{i_1}\right)$$

por lo que también vale cero si $\check{q_{i_1}} = b_{i_1}$. Es decir,

$$f_{x_{i_\theta}}(q) = 0 \ \ \forall \theta \in \{2, 3, ..., m\}$$

Análogamente, tomando ahora

$$\dot{q} = (q_1, ..., b_{i_1}, ..., q_{i_2-1}, \dot{q_{i_2}}, q_{i_2+1}, ..., b_{i_3}, ..., b_{i_m}, ..., q_n, t_1) \text{ con } \dot{q_{i_2}} \in \left(\frac{a_{i_2} + b_{i_2}}{2}, b_{i_2}\right)$$

se demuestra que $f_{x_{i_1}}(q) = 0$.

Por consiguiente, obtenemos que

$$f_{x_i}(q) = 0 \quad \forall i : \ q_i \in \{a_i, b_i\}$$

Para el caso en que $q = (q_1, q_2, ..., q_n, b) \in \partial_S D$, dado cualquier $i \in \{1, 2, ..., n\}$ tal que $q_i \in \{a_i, b_i\}$, tenemos que $f_{x_i}(q_1, q_2, ..., q_n, t_2) = 0 \ \ \forall t_2 \in (a, b)$. Como f_{x_i} es continua, $f_{x_i}(q) = 0$.

Con esto, podemos demostrar que $\nabla f(p) = \big(f_{x_1}(p), f_{x_2}(p), ..., f_{x_n}(p)\big) = (0, 0, ..., 0)$. Para ello, separaremos la prueba en dos partes:

- Dado $i \in \{1, 2, ..., n\}$ tal que $p_i \in \{a_i, b_i\}$, entonces, como acabamos de demostrar, $f_{x_i}(p) = 0$.
- Dado $i \in \{1, 2, ..., n\}$ tal que $p_i \in (a_i, b_i)$, se tiene que $f(p) \geq f(p_1, p_2, ..., p_{i-1}, \gamma, p_{i+1}, ..., p_n, t_0)$ $\forall \gamma \in (a_i, b_i)$. Por tanto, restringiéndonos a la dirección del eje x_i, el punto p es un extremo relativo; por lo que $f_{x_i}(p) = 0$.

Por otro lado, también podemos probar que $f_t(p) \geq 0$:

- Si $t_0 \in (a, b)$, al restringirnos a la dirección del eje t, el punto p es un extremo relativo, por lo que $f_t(p) = 0$.
- Si $t_0 = b$, se demuestra que $f_t(p) \geq 0$ de manera idéntica a como se demostró en el **Teorema 1**, cuando supusimos que $p \in \partial_T D$.

Con todo esto, podemos proceder como en el **Teorema 1**, suponiendo que $C(x,t)\cdot f_t - c(x,t)\cdot\Delta f - \nabla c\cdot\nabla f < 0$, pero construiremos n polinomios de Taylor de grado 2, eligiendo en cada caso un vector $v_i \in \mathbb{R}^{n+1}$ conveniente.
Para cada $i \in \{1,2,...,n\}$, definimos el vector v_i como:

$$v_i = \begin{cases} (0,0,...,0,\underset{i)}{1},0,...,0) & \text{si } p_i \in [a_i,b_i) \\ \\ (0,0,...,0,\underset{i)}{-1},0,...,0) & \text{si } p_i = b_i \end{cases}$$

De esta manera, para cada v_i, existirá un $\varepsilon_i > 0$ suficientemente pequeño, tal que

$$p + \varepsilon_i \cdot v_i \in \overline{D} \quad \forall i \in \{1,2,...,n\}$$

Entonces:

$$f(p+\varepsilon_i\cdot v_i) = f(p) + \varepsilon_i\cdot\nabla f^T(p)\cdot v_i + \frac{\varepsilon_i^2}{2}\cdot v_i^T\cdot\mathcal{H}f(p)\cdot v_i + \mathrm{O}(\varepsilon_i^3) \leq f(p)$$

Como ya hemos visto, $\nabla f(p) = (0,0,...,0)$. Por tanto

$$f(p) + \frac{\varepsilon_i^2}{2}\cdot v_i^T\cdot\mathcal{H}f(p)\cdot v_i + \mathrm{O}(\varepsilon^3) \leq f(p) \Longrightarrow \frac{\varepsilon_i^2}{2}\cdot v_i^T\cdot\mathcal{H}f(p)\cdot v_i \leq 0$$

Independientemente de lo que valga v_i:

$$(0,0,...,0,\underset{i)}{1},0,...,0)\cdot\mathcal{H}f(p)\cdot\begin{pmatrix}0\\0\\\vdots\\0\\1\\0\\\vdots\\0\end{pmatrix}_{i)} = \left(\frac{\partial^2 f(p)}{\partial x_i x_1},\frac{\partial^2 f(p)}{\partial x_i x_2},...,\frac{\partial^2 f(p)}{\partial x_i x_n},\frac{\partial^2 f(p)}{\partial x_i x_t}\right)\cdot\begin{pmatrix}0\\0\\\vdots\\0\\1\\0\\\vdots\\0\end{pmatrix}_{i)} = \frac{\partial^2 f(p)}{\partial x_i^2}$$

$$(0,0,...,0,\underset{i)}{-1},0,...,0)\cdot\mathcal{H}f(p)\cdot\begin{pmatrix}0\\0\\\vdots\\0\\-1\\0\\\vdots\\0\end{pmatrix}_{i)}=$$

$$=\big(-\frac{\partial^2 f(p)}{\partial x_i x_1},-\frac{\partial^2 f(p)}{\partial x_i x_2},...,-\frac{\partial^2 f(p)}{\partial x_i x_n},-\frac{\partial^2 f(p)}{\partial x_i x_t}\big)\cdot\begin{pmatrix}0\\0\\\vdots\\0\\-1\\0\\\vdots\\0\end{pmatrix}_{i)}=\frac{\partial^2 f(p)}{\partial x_i^2}$$

en cualquier caso, obtenemos que

$$\frac{\varepsilon_i^2}{2}\cdot v_i^T\cdot\mathcal{H}f(p)\cdot v_i=\frac{\varepsilon_i^2}{2}\cdot\frac{\partial^2 f(p)}{\partial x_i^2}\leq 0$$

es decir,

$$\frac{\partial^2 f(p)}{\partial x_i^2}\leq 0\quad\forall i\in\{1,2,...,n\}$$

Por tanto, tenemos que

$$\Delta f(p)=\frac{\partial^2 f(p)}{\partial x_1^2}+\frac{\partial^2 f(p)}{\partial x_2^2}+...+\frac{\partial^2 f(p)}{\partial x_n^2}\leq 0$$

En resumen, sabemos que $C(p)$, $c(p)$, $f_t(p)\geq 0$; que $\nabla f(p)=0$, y que $\Delta f(p)\leq 0$.

Con todo esto, llegamos a que

$$C(p)\cdot f_t(p)-c(p)\cdot\Delta f(p)-\nabla c(p)\cdot\nabla f(p)=C(p)\cdot f_t(p)-c(p)\cdot\Delta f(p)\geq 0$$

llegando a una contradicción, y concluyendo así este caso.

► En el caso más general $C(x,t)\cdot f_t-c(x,t)\cdot\Delta f-\nabla c\cdot\nabla f\leq 0$, se procede de manera análoga al caso en el que $p\in D$ (segunda parte del **Teorema 1**).

Por lo tanto, queda demostrado que

$$p=\max_{\overline{D}} f=\max_{\partial_{SB}D} f\notin\partial_S D$$

es decir,

$$\max_{\overline{D}} f \in \partial_B D$$

□

Para el caso que nos ocupa, tomando $C(x,t) = 1$ y $f = I$ obtenemos que la ecuación (3) satisface el **Teorema 1**. Si además suponemos que cumple las condiciones de contorno de Neumann antes mencionadas, tenemos que el punto donde la función I alcanza el máximo se encuentra en la imagen inicial.

Para ver que el mínimo de I también pertenece a la imagen inicial, tomaremos la función $h = -I$, que es igual de diferenciable que I, y cumple que $h_t = -I_t$, $\nabla h = -\nabla I$ y $\Delta h = -\Delta I$.
Entonces

$$h_t = -I_t = -c(x,y,t) \cdot \Delta I - \nabla c \cdot \nabla h = c(x,y,t) \cdot \Delta h + \nabla c \cdot \nabla h$$

es decir, h satisface el **Teorema 1**.

Además, dado cualquier punto $(x_1, x_2, ..., x_n, t) \in \overset{\sim}{\partial_S} D$, se cumple que

$$\begin{aligned}(\nabla h, h_t)(x_1, x_2, ..., x_n, t) \cdot u(x_1, x_2, ..., x_n, t) = (-\nabla I, -I_t)(x_1, x_2, ..., x_n, t) \cdot u(x_1, x_2, ..., x_n, t) = \\ = -(\nabla I, I_t)(x_1, x_2, ..., x_n, t) \cdot u(x_1, x_2, ..., x_n, t) = 0\end{aligned}$$

Por tanto, h también cumple las condiciones de contorno de Neumann sobre $\overset{\sim}{\partial_S} D$ y, en consecuencia, satisface el **Corolario 1**.

Finalmente, como el máximo de h es precisamente el mínimo de I, concluimos que el punto donde la función I alcanza el mínimo también se encuentra en la imagen inicial.

Con esto, se verifica que la función I satisface la condición de causalidad.

2.3. Realce de los bordes

Como ya hemos visto, nuestro objetivo es conseguir aplicar un filtro a la imagen con el que se resalten los bordes de ésta, y no se mezclen con sus distintas regiones. Para ello, tendremos que elegir una función $c(x,y,t)$ adecuada, que consiga que la pendiente en los bordes se mantenga, o aumente cuando aumenta t.

Para simplificar el estudio, y sin pérdida de generalidad, podemos suponer que los bordes son paralelos al eje y, con lo que nos queda un problema en una dimensión.

2.3.1. Concepto de borde

En esta sección, trataremos de formalizar la idea intuitiva de borde, para así poder estudiar más rigurosamente la manera de realzarlos.

Según Nelson Bahamón [2] "Al referirse a bordes se puede pensar que lo que se está buscando son contornos, lo cual no es necesariamente cierto. En sí el propósito de los algoritmos de detección de bordes es obtener como resultado una imagen donde se resalten los pixeles de aquellos puntos de la imagen original en donde se presentan cambios bruscos de intensidad".
El siguiente es un ejemplo de una imagen cuyos contornos no coinciden con sus bordes, y la función asociada al recorrer la imagen horizontalmente en la mitad de su altura:

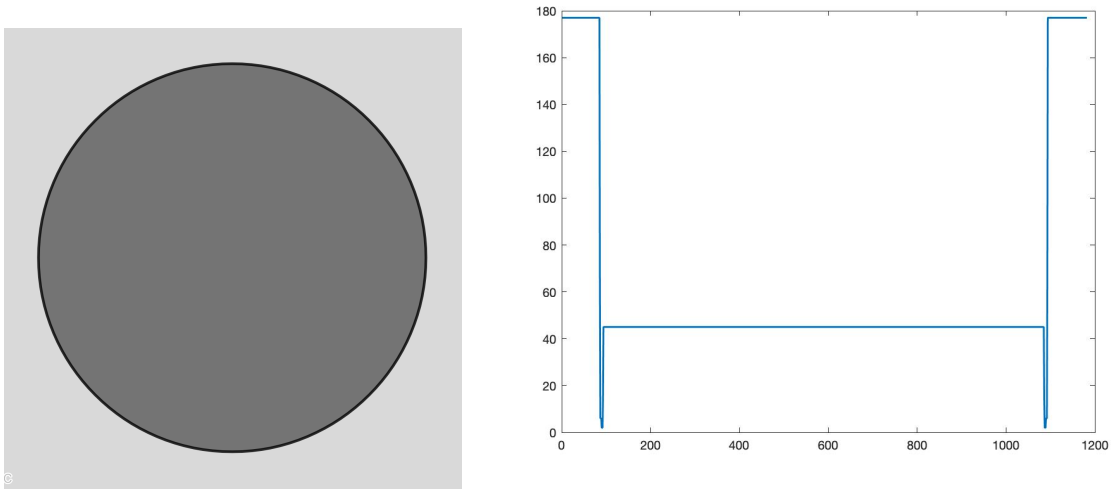

Figura 6: Imagen de un círculo construido con GeoGebra Classic 6 y su representación al recorrerla horizontalmente a la mitad de su altura construida con Matlab 2016b.

El contorno de la imagen sería la circunferencia, mientras que los bordes serían las partes interior y exterior de esta.

Valiéndonos de la derivada de la función anterior, resulta más fácil localizar los bordes:

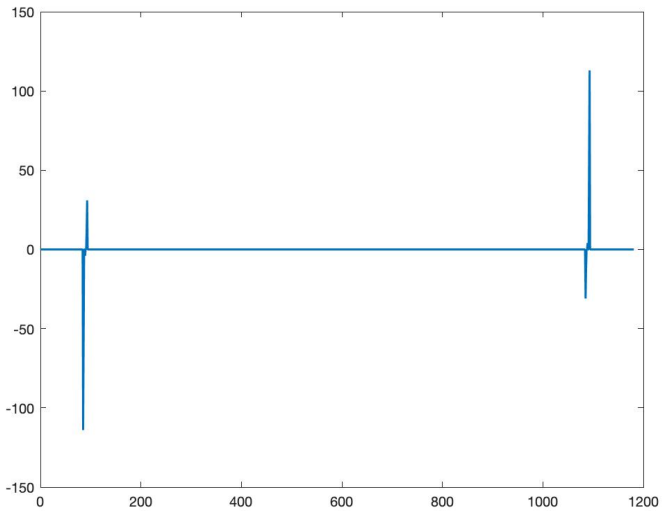

Figura 7: Gráfica de la derivada de la función anterior, construida con Matlab 2016b.

Se aprecian cuatro extremos relativos, que se corresponderán con los cuatro bordes; dos por cada vez que se recorre horizontalmente el contorno.

Intuitivamente, un borde sería todo punto donde el valor absoluto de la derivada tiene un máximo relativo; pero, además, será necesario que el valor absoluto de la derivada en estos puntos tome valores suficientemente grandes para ser considerados bordes.

Por ejemplo, dadas las dos imágenes siguientes, al recorrerlas horizontalmente en cualquier altura, la de la izquierda debería tener 6 bordes, mientras que la de la derecha no debería tener ninguno.

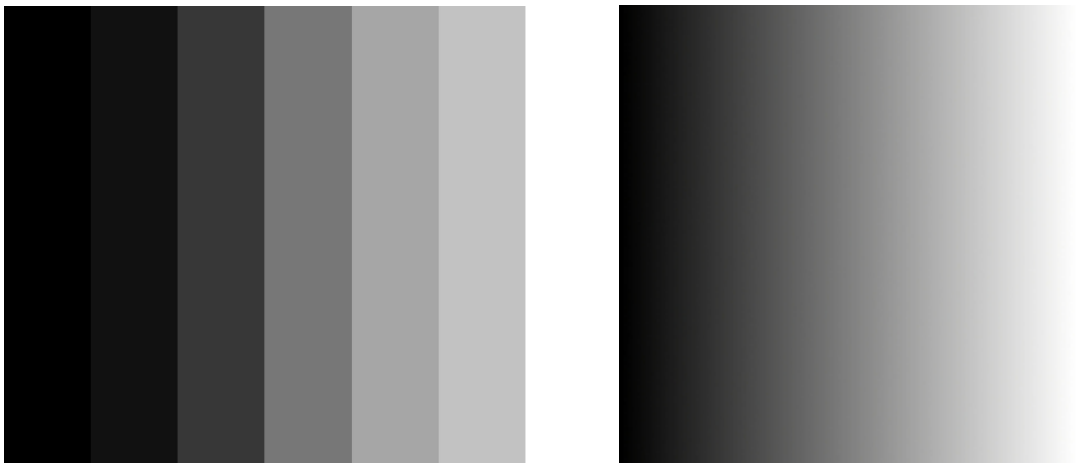

Figura 8: Dos escalas de grises distintas: la de la izquierda, construida con Paint, son seis franjas bien definidas con distintos niveles, mientras que en la de la derecha, tomada de [1], el cambio es progresivo.

Sus gráficas serían:

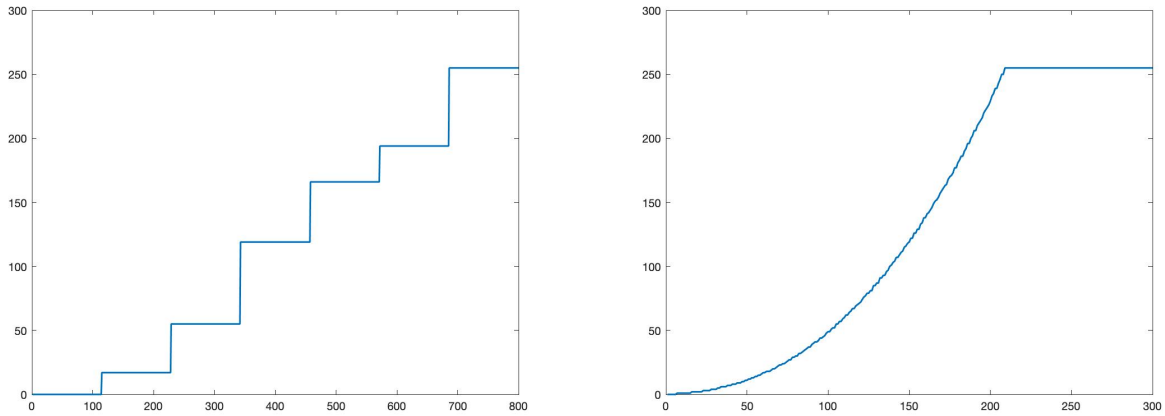

Figura 9: Gráficas para comparar las imagenes de la Figura anterior, construidas con Matlab 2016b.

y las gráficas de sus derivadas serían:

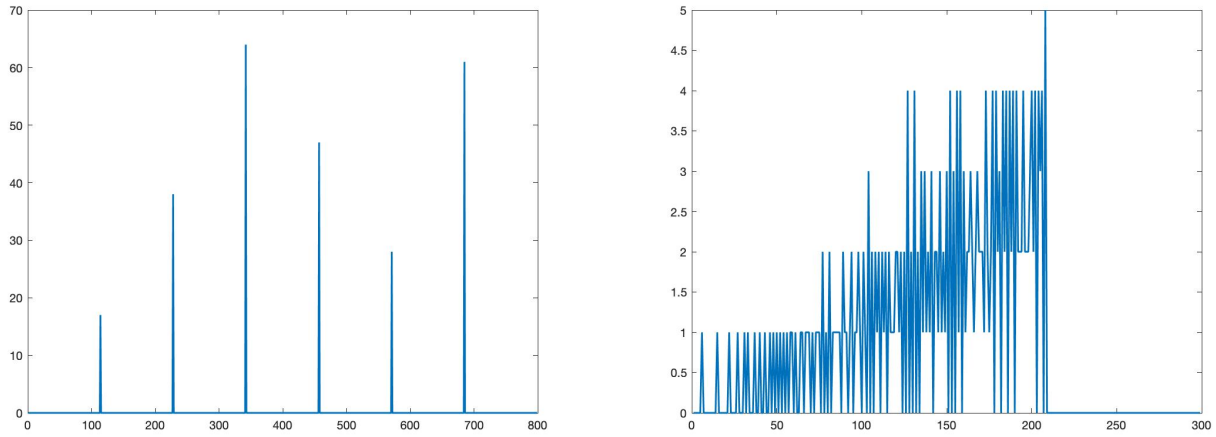

Figura 10: Gráficas de las derivadas, construidas con Matlab 2016b.

A partir de la derivada, se ven claramente los 6 bordes de la imagen izquierda, mientras que en la derecha, aunque su derivada posee muchos máximos y mínimos relativos, estos no tienen suficiente valor absoluto como para considerarse bordes. Por tanto, habrá que introducir un término λ que determine la pendiente mínima que deben tener los bordes.

De esta manera, siendo $Dom(I)$ el dominio de I, es decir, las dimensiones de la imagen; podemos definir el conjunto B de bordes de I, donde I es vista como una función unidimensional, de la siguiente manera:

$$B = \left\{ x \in Dom(I) : \exists\, E \text{ entorno de } x, \text{ tal que } \max_{y \in E} (|I'(y)|) = |I'(x)| \ \wedge \ |I'(x)| \geq \lambda \right\}$$

Si la imagen I tiene dos dimensiones, la definición sería:

$$B = \left\{ x \in Dom(I) : \exists\, E \text{ entorno de } x, \text{ tal que } \max_{y \in E} \big(||\nabla I(y)||\big) = ||\nabla I(x)|| \ \wedge \ ||\nabla I(x)|| \geq \lambda \right\}$$

Es decir, cuanto mayor sea λ, menos bordes habrá. Empíricamente, parece razonable tomar un valor alrededor de 3 o 6. Sin embargo, existen métodos más precisos para el cálculo de este valor, que dependen de la imagen, y por tanto, λ pasará a depender de la variable t, como tomar $\lambda = \frac{\max \|\nabla I\|}{10}$, es decir, una décima parte del máximo gradiente de la imagen. También existen métodos más complejos, como el propuesto por Canny [3].

2.3.2. Fundamentos matemáticos del realce

Atendiendo a la ecuación (3), en una dimensión, quedará:

$$I_t = div\big(c(x,t) \cdot \nabla I\big) = \frac{\partial}{\partial x}\big(c(x,t) \cdot I_x\big)$$

donde ya habíamos determinado que la función c será de la forma $c(x,t) = g\big(I_x(x,t)\big)$. Por tanto, durante toda esta sección, para simplificar la notación escribiremos $g(I_x)$ al referirnos a esta función.

Una vez elegido un valor de λ determinado, la función g se tendrá que ajustar de tal forma que el filtro realmente conserve los bordes y suavice las regiones limitadas por estos. Para ello, se propondrán dos familias de funciones y se analizará qué funciones de cada familia son válidas para cada λ.

Sea $\phi(I_x) := g(I_x) \cdot I_x$, por la regla de la cadena, obtenemos que:

$$I_t = \frac{\partial}{\partial x}(\phi(I_x)) = \phi'(I_x) \cdot I_{xx} \tag{5}$$

es una versión en una dimensión de la ecuación de difusión.

El incremento de la pendiente con respecto al tiempo vendrá dado por $\frac{\partial}{\partial t}(I_x)$; y como suponemos que la función es suficientemente regular, podremos cambiar el orden de derivación, para obtener:

$$\frac{\partial}{\partial t}(I_x) = \frac{\partial}{\partial x}(I_t) = \frac{\partial}{\partial x}\Big(\frac{\partial}{\partial x}\big(\phi(I_x)\big)\Big) = \phi''(I_x) \cdot I_{xx}^2 + \phi'(I_x) \cdot I_{xxx} \tag{6}$$

Ahora analizaremos dos posibles casos:

- Primero, supondremos que en el borde se tiene que $I_x > 0$, es decir, que la imagen es más oscura a la izquierda del borde que a la derecha. Entonces, en ese punto, $I_{xx} = 0$ y $I_{xxx} < 0$ por ser un máximo relativo de la derivada. Por tanto:

  $$\frac{\partial}{\partial t}(I_x) = \phi''(I_x) \cdot I_{xx}^2 + \phi'(I_x) \cdot I_{xxx} = \phi'(I_x) \cdot I_{xxx} = -\phi'(I_x) \cdot |I_{xxx}|$$

  Es decir, en un entorno del borde, $\frac{\partial}{\partial t}(I_x)$ tiene signo contrario a $\phi'(I_x)$. Esto significa que si $\phi'(I_x) > 0$, la función I_x decrecerá, y la pendiente del borde disminuirá con el tiempo, mientras que si $\phi'(I_x) < 0$, la función I_x crecerá, por lo que la pendiente aumentará. Cabe destacar que por el principio de causalidad, un incremento en la pendiente no puede ser debido a un aumento en la diferencia de intensidad entre píxeles. El motivo deberá ser que los bordes se hacen más "finos", es decir, más "agudos".

- Cuando $I_x < 0$ en el borde, se tiene que $I_{xx} = 0$ y $I_{xxx} > 0$. Entonces:

  $$\frac{\partial}{\partial t}(I_x) = \phi''(I_x) \cdot I_{xx}^2 + \phi'(I_x) \cdot I_{xxx} = \phi'(I_x) \cdot I_{xxx} = \phi'(I_x) \cdot |I_{xxx}|$$

  Por lo tanto, en un entorno del borde, $\frac{\partial}{\partial t}(I_x)$ tiene el mismo signo que $\phi'(I_x)$. Esto significa que si $\phi'(I_x) > 0$, la función I_x crecerá; es decir, decrecerá en valor absoluto, por lo que la pendiente del borde disminuirá con el tiempo, mientras que si $\phi'(I_x) < 0$, la función I_x decrecerá; es decir, crecerá en valor absoluto, por lo que la pendiente en el borde aumentará.

Como ya se ha explicado en la **Sección 2.1**, la función g deberá cumplir ciertas propiedades. En este trabajo se estudiarán dos familias de funciones que satisfacen todos los requisitos necesarios:

• Las primeras serían las funciones racionales de la forma

$$g(I_x) = \frac{C}{1 + \left(\frac{\|I_x\|}{K}\right)^{1+\alpha}}$$

con $C, \alpha, K > 0$ constantes a determinar.

Dado que nos vamos a centrar en estudiar la función $\phi'(I_x)$, la constante C no tiene mucha relevancia, así que podremos tomar, por ejemplo $C = 1$. Tras derivar, obtenemos:

$$\phi'(I_x) = K \cdot \frac{K - \alpha \cdot ||I_x|| \cdot \left(\frac{||I_x||}{K}\right)^{\alpha}}{\left(K + ||I_x|| \cdot \left(\frac{||I_x||}{K}\right)^{\alpha}\right)^2}$$

y esta función se anulará si y solo si $K - \alpha \cdot ||I_x|| \cdot \left(\frac{||I_x||}{K}\right)^{\alpha} = 0$, es decir, si y solo si $||I_x|| = \frac{K}{\alpha+\sqrt[1]{\alpha}}$.

Entonces, cuando $I_x \in (0, \frac{K}{\alpha+\sqrt[1]{\alpha}})$, tenemos que $\phi'(I_x) > 0$ y, como hemos visto, la pendiente decrecerá, mientras que si $I_x \in (\frac{K}{\alpha+\sqrt[1]{\alpha}}, +\infty)$, tenemos que $\phi'(I_x) < 0$ y el borde se intensificará.

Por otro lado, si $I_x \in (-\infty, -\frac{K}{\alpha+\sqrt[1]{\alpha}})$, tenemos que $\phi'(I_x) > 0$ y entonces la pendiente decrecerá, mientras que si $I_x \in (-\frac{K}{\alpha+\sqrt[1]{\alpha}}, 0)$, tenemos que $\phi'(I_x) < 0$ y la pendiente aumentará.

Es decir, la pendiente en los bordes donde $||I_x|| < \frac{K}{\alpha+\sqrt[1]{\alpha}}$ se reducirá, y en los que $||I_x|| > \frac{K}{\alpha+\sqrt[1]{\alpha}}$, aumentará.
Si $||I_x|| = \frac{K}{\alpha+\sqrt[1]{\alpha}}$, se tiene que $\phi'(I_x) = 0$, por lo que la pendiente en el borde no cambiará.

Por consiguiente, el valor de $\frac{K}{\alpha+\sqrt[1]{\alpha}}$ deberá ser el que consideremos suficiente para diferenciar un borde de una impureza; es decir, tendremos que tomar K y α tales que

$$\lambda = \frac{K}{\alpha+\sqrt[1]{\alpha}}$$

Por ejemplo, tomando $\alpha = 1$, y $K = \lambda$. De este modo, los bordes, donde $||I_x|| > \lambda$, se remarcarán, y las imperfecciones, donde $||I_x|| < \lambda$, se suavizarán.

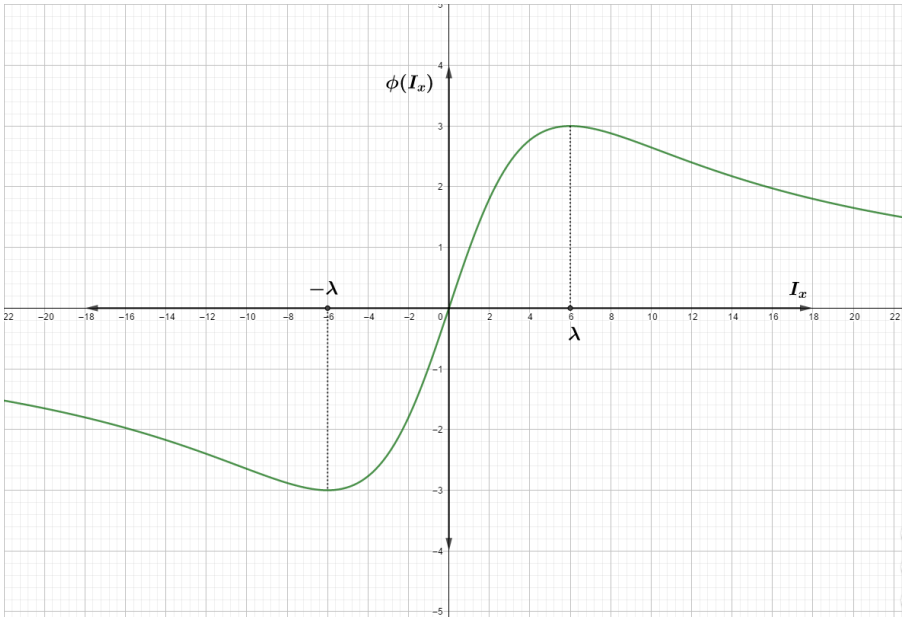

Figura 11: Ejemplo de una posible función ϕ donde $C = 1$, $\alpha = 1$ y $K = 6$, y por tanto $\lambda = 6$. Gráfica construida con GeoGebra Classic 6.

• Otra posible idea sería tomar una función exponencial, de la forma

$$g(I_x) = e^{-\left(\frac{\|I_x\|}{K}\right)^2}$$

En este caso:

$$\phi'(I_x) = \frac{e^{-\frac{||I_x||^2}{K^2}} \cdot (K^2 - 2 \cdot ||I_x||^2)}{K^2}$$

Razonando igual que antes, obtenemos que habría que tomar $K = \lambda\sqrt{2}$.

Para ambas funciones, como se señala en [8], tomar un λ demasiado pequeño provoca que el suavizado se reduzca y el proceso sea más lento, mientras que tomarlo demasiado grande lo acelera, pero a riesgo de perder información de los bordes, como en el filtro gaussiano, que es precisamente lo que se pretende evitar.

3. Implementación de filtros en imágenes unidimensionales

Para poder aplicar todo lo estudiado anteriormente a casos reales, se deberán tratar de discretizar todas las ideas que, en principio, estaban pensadas sobre un marco de matemática continua, para así computarlas y llevarlas a la práctica.

Inicialmente, trataremos imágenes en una dimensión debido a su menor dificultad, y después pasaremos a las dos dimensiones.

La imagen que vamos a tomar de ejemplo, y con la que vamos a trabajar es la siguiente:

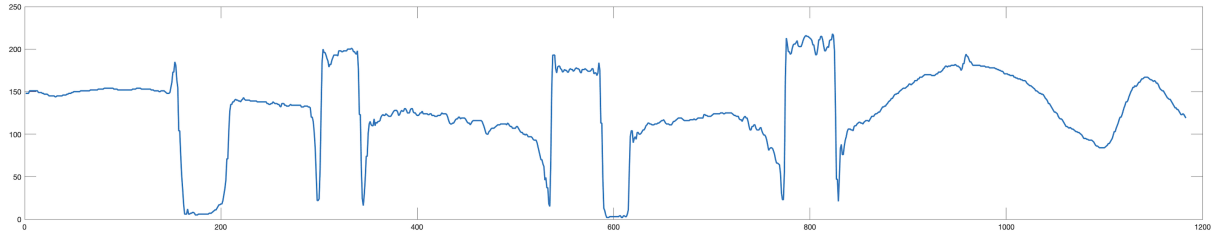

Figura 12: Imagen unidimensional obtenida al recorrer horizontalmente, a mitad de altura, la foto original del pez payaso de la Figura 2. Imagen obtenida con Matlab 2016b.

Como se aprecia a simple vista, esta imagen tiene muchos bordes, bien marcados, y también muchas partes "rugosas" que pretenderemos suavizar.

Lo primero será dar una definición formal de derivada discreta. Para ello, estudiaremos la definición estándar de derivada:

Sea $f : \mathbb{R} \longrightarrow \mathbb{R}$ una función, se define la derivada de f en un punto $z \in \mathbb{R}$ como:

$$f_x(z) = \lim_{h \to 0} \frac{f(z+h) - f(z)}{h}$$

siempre que el límite exista.

Si dicho límite existe, se tiene que

$$\lim_{h \to 0} \frac{f(z+h) - f(z)}{h} = \lim_{h \to 0^+} \frac{f(z+h) - f(z)}{h} = \lim_{h \to 0^-} \frac{f(z+h) - f(z)}{h}$$

es decir, existe el límite al acercarse por ambos lados del punto, y es el mismo.

Sea $I : \{0, 1, 2, ..., n\} \to \mathbb{R}$ una función unidimensional, y sea $x_0 \in \{0, 1, 2, ..., n\}$, una posible discretización de estos límites sería:

$$\lim_{h \to 0^+} \frac{I(x_0+h) - I(x_0)}{h} \approx \frac{I(x_0+1) - I(x_0)}{x_0 + 1 - x_0} = I(x_0+1) - I(x_0)$$

$$\lim_{h \to 0^-} \frac{I(x_0+h) - I(x_0)}{h} \approx \frac{I(x_0-1) - I(x_0)}{x_0 - 1 - x_0} = I(x_0) - I(x_0-1)$$

puesto que $x_0 + 1$ y $x_0 - 1$ son los puntos más cercanos a x_0.
Para los casos particulares $x_0 = 0$ y $x_0 = n$, dado que I deberá cumplir las condiciones de contorno de Neumann, supondremos que $I(-1) = I(0)$ y $I(n+1) = I(n)$.
Nótese que le hemos dado valor 1 a la distancia entre puntos contiguos; es decir, hemos tomado $\delta x = 1$.

De esta manera, en principio, podríamos hablar de dos derivadas discretas que, en un marco continuo, deberían coincidir:

$$I_{x^+}(x_0) := I(x_0+1) - I(x_0)$$

$$I_{x^-}(x_0) := I(x_0) - I(x_0-1)$$

Por ello, puesto que la función que realmente necesitaremos discretizar es la segunda derivada, utilizaremos ambas definiciones para su construcción:

$$\begin{aligned} I_{xx}(x_0) =& (I_{x^+})_{x^-}(x_0) \\ =& I_{x^+}(x_0) - I_{x^+}(x_0 - 1) \\ =& I(x_0 + 1) - I(x_0) - \big(I(x_0) - I(x_0 - 1)\big) \\ =& I(x_0 + 1) + I(x_0 - 1) - 2 \cdot I(x_0) \end{aligned}$$

Es fácil comprobar que, independientemente del orden en que derivemos, el resultado es el mismo.
De esta manera, ambas derivadas tendrán la misma importancia a la hora de obtener la segunda derivada, y esta se podrá calcular en cada punto conociendo únicamente el valor de la función en el punto, y en los puntos contiguos.

Nuestro objetivo será construir una secuencia de imágenes unidimensionales, $I^0, I^1, I^2, ..., I^{t_0}$ a partir de la inicial, I^0, hasta obtener una final I^{t_0}, donde t_0 es un número de iteraciones determinado.

3.1. Implementación del Filtro Gaussiano unidimensional

Como ya hemos visto, la ecuación utilizada en el filtro gaussiano es $I_t = \Delta I$.
La derivada respecto al tiempo en un punto x_0, se discretizará como

$$I_t(x_0) = \frac{I^{t+1}(x_0) - I^t(x_0)}{\delta t}$$

donde δt es el intervalo de tiempo entre dos iteraciones consecutivas.

Como veremos en la **Sección 3.2**, para asegurar la convergencia del método, será necesario que el Número de Courant–Friedrichs–Lewy, $\Lambda := \frac{\delta t}{\delta x^2}$, sea menor que $\frac{1}{2}$.

Dado que hemos tomado $\delta x = 1$ (la distancia entre dos puntos contiguos, x_0 y $x_0 + 1$, será $(x_0 + 1) - (x_0) = 1$), es suficiente con tomar $\delta t \in (0, \frac{1}{2}]$. Tomaremos, por ejemplo, $\delta t = \frac{1}{4}$, y, entonces, construimos la secuencia

$$I^{t+1}(x_0) = I^t(x_0) + \frac{1}{4} \cdot (I^t(x_0 + 1) + I^t(x_0 - 1) - 2 \cdot I^t(x_0))$$

Ejecutando el algoritmo para distintas iteraciones, obtenemos:

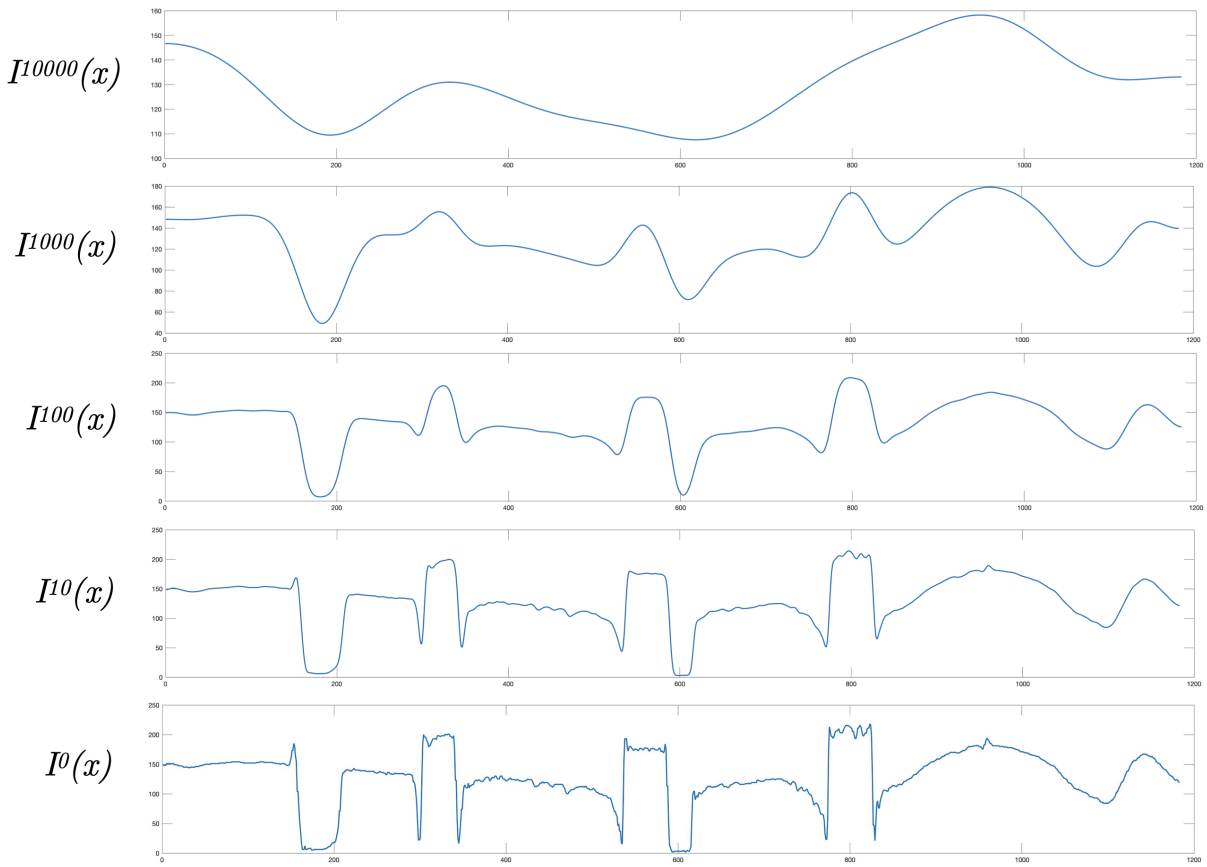

Figura 13: Imágenes obtenidas tras aplicar el filtro gaussiano 0, 10, 100, 1000 y 10000 veces. Gráficas obtenidas con Matlab 2016b.

Aquí se aprecia claramente el problema del filtro gaussiano del que ya hemos hablado. El suavizado se realiza de igual forma en todas las regiones, ocasionando una pérdida de significado en la imagen cuando el número de iteraciones es suficientemente grande. En este caso, para $t = 10$, el suavizado aún no es completo, como se aprecia en algunas regiones (desde 400 a 450, ó desde 650 a 700, por ejemplo), pero ya se ha reducido significativamente la intensidad de algunos bordes, como los de 300 y 350. Con $t = 100$ esos bordes ya son prácticamente inexistentes, aunque todavía quedan regiones sin suavizar completamente; y en $t = 1000$ ya se han perdido la mayoría de los bordes. Con $t = 10000$, la imagen ya ha perdido todo su significado.

Por esta razón, implementaremos el filtro de Perona-Malik, y lo aplicaremos a esta imagen para compararlo con el filtro gaussiano.

3.2. Implementación del Filtro de Perona-Malik unidimensional

La ecuación de Perona-Malik, en una dimensión, se reduce a

$$I_t(x_0, t) = \frac{\partial((c \cdot I_x)(x_0, t))}{\partial x} = \frac{\partial\Big(g\big(|I_x(x_0, t)|\big) \cdot I_x(x_0, t)\Big)}{\partial x}$$

De manera similar a como se ha visto en el filtro gaussiano, la ecuación se discretizará como:

$$\begin{aligned}\frac{I^{t+1}(x_0)-I^t(x_0)}{\delta t} &= \Big(g\big(|I^t_{x^+}|\big)\cdot I^t_{x^+}\Big)_{x^-}(x_0)\\ &= \Big(g\big(|I^t_{x^+}|\big)\cdot I^t_{x^+}\Big)(x_0)-\Big(g\big(|I^t_{x^+}|\big)\cdot I^t_{x^+}\Big)(x_0-1)\\ &= g\big(|I^t(x_0+1)-I^t(x_0)|\big)\cdot\big(I^t(x_0+1)-I^t(x_0)\big)\\ &- g\big(|I^t(x_0)-I^t(x_0-1)|\big)\cdot\big(I^t(x_0)-I^t(x_0-1)\big) \qquad (7)\end{aligned}$$

Ahora demostraremos un teorema que nos asegura la convergencia del método cuando tomamos un paso temporal en $(0,\frac{1}{2}]$:

Teorema 2 (**Principio del Máximo y del Mínimo Discretos en una dimensión**). *Sea $n\in\mathbb{N}$. Dada una secuencia de funciones $I^0, I^1, ..., I^{t_0}$, donde $I^t : \{1,2,...,n\}\to\{0,1,...,255\}$ $\forall t\in\{0,1,...,t_0\}$, que satisface la ecuación (7) cuando $0<\delta t\leq\frac{1}{2}$ y $g:\mathbb{R}^+\to[0,1]$; se tiene que el valor máximo y el mínimo de entre todas las funciones lo alcanza I^0, es decir, siendo $I^t_m=\min_{x_0\in\{1,2,...,n\}}\big(I^t(x_0)\big)$, y $I^t_M=\max_{x_0\in\{1,2,...,n\}}\big(I^t(x_0)\big)$, se cumple que:*

$$I^0_m\leq I^t(x)\leq I^0_M \quad \forall t\in\{0,1,...,t_0\},\ \forall x\in\{1,2,...,n\}$$

Demostración:

La demostración se basará en probar que $I^t_m\leq I^{t+1}(x)\leq I^t_M \quad \forall x\in\{1,2,...,n\}$.

Dado cualquier $x_0\in\{1,2,...,n\}$, a partir de la ecuación (7):

$$\begin{aligned}I^{t+1}(x_0) = I^t(x_0)+\delta t\cdot\Big(&g\big(|I^t(x_0+1)-I^t(x_0)|\big)\cdot\big(I^t(x_0+1)-I^t(x_0)\big)+\\ &+g\big(|I^t(x_0)-I^t(x_0-1)|\big)\cdot\big(I^t(x_0-1)-I^t(x_0)\big)\Big)=\\ = I^t(x_0)\cdot\Big(1-\delta t\cdot\Big(&g\big(|I^t(x_0+1)-I^t(x_0)|\big)+g\big(|I^t(x_0)-I^t(x_0-1)|\big)\Big)\Big)+\\ +\delta t\cdot\Big(&g\big(|I^t(x_0+1)-I^t(x_0)|\big)\cdot I^t(x_0+1)+g\big(|I^t(x_0)-I^t(x_0-1)|\big)\cdot I^t(x_0-1)\Big)\end{aligned}$$

Para simplificar la notación, llamaremos $g_1=g\big(|I^t(x_0+1)-I^t(x_0)|\big)$,y $g_2=g\big(|I^t(x_0)-I^t(x_0-1)|\big)$.
Como g_1, $g_2\in[0,1]$, y $0<\delta t\leq\frac{1}{2}$, se tiene que

$$1-\delta t\cdot(g_1+g_2)\geq 0$$

Entonces, deducimos que

$$\begin{aligned}I^{t+1}(x_0)&\leq I^t_M\cdot\big(1-\delta t\cdot(g_1+g_2)\big)+\delta t\cdot(g_1\cdot I^t_M+g_2\cdot I^t_M)=\\ &= I^t_M-I^t_M\cdot\delta t\cdot(g_1+g_2)+I^t_M\cdot\delta t\cdot(g_1+g_2)=I^t_M\end{aligned}$$

en particular,

$$I_M^0 \geq I_M^1 \geq I_M^2 \geq ... \geq I_M^{t_0}$$

es decir

$$I_M^0 \geq I^t(x_0) \quad \forall t \in \{0, 1, ..., t_0\},\ \forall x_0 \in \{1, 2, ..., n\}$$

De manera análoga:

$$\begin{aligned} I^{t+1}(x_0) &\geq I_m^t \cdot \big(1 - \delta t \cdot (g_1 + g_2)\big) + \delta t \cdot (g_1 \cdot I_m^t + g_2 \cdot I_m^t) = \\ &= I_m^t - I_m^t \cdot \delta t \cdot (g_1 + g_2) + I_m^t \cdot \delta t \cdot (g_1 + g_2) = I_m^t \end{aligned}$$

y por tanto

$$I_m^0 \leq I^t(x_0) \quad \forall t \in \{0, 1, ..., t_0\},\ \forall x_0 \in \{1, 2, ..., n\}$$

□

Nótese que para el filtro gaussiano, la ecuación utilizada es un caso particular de (7) en el que $g \equiv 1$, y, por tanto, satisface el Principio del Máximo y Mínimo Discretos cuando $\delta t \in (0, \frac{1}{2}]$.

Finalmente, como ya hemos visto, probaremos con diferentes funciones $g(|I_x|)$ y las iremos evaluando y comparando.

Tomando $g(|I_x|) = \frac{1}{1+\left(\frac{|I_x|}{4}\right)^2}$ y ejecutando el algoritmo para distintas iteraciones:

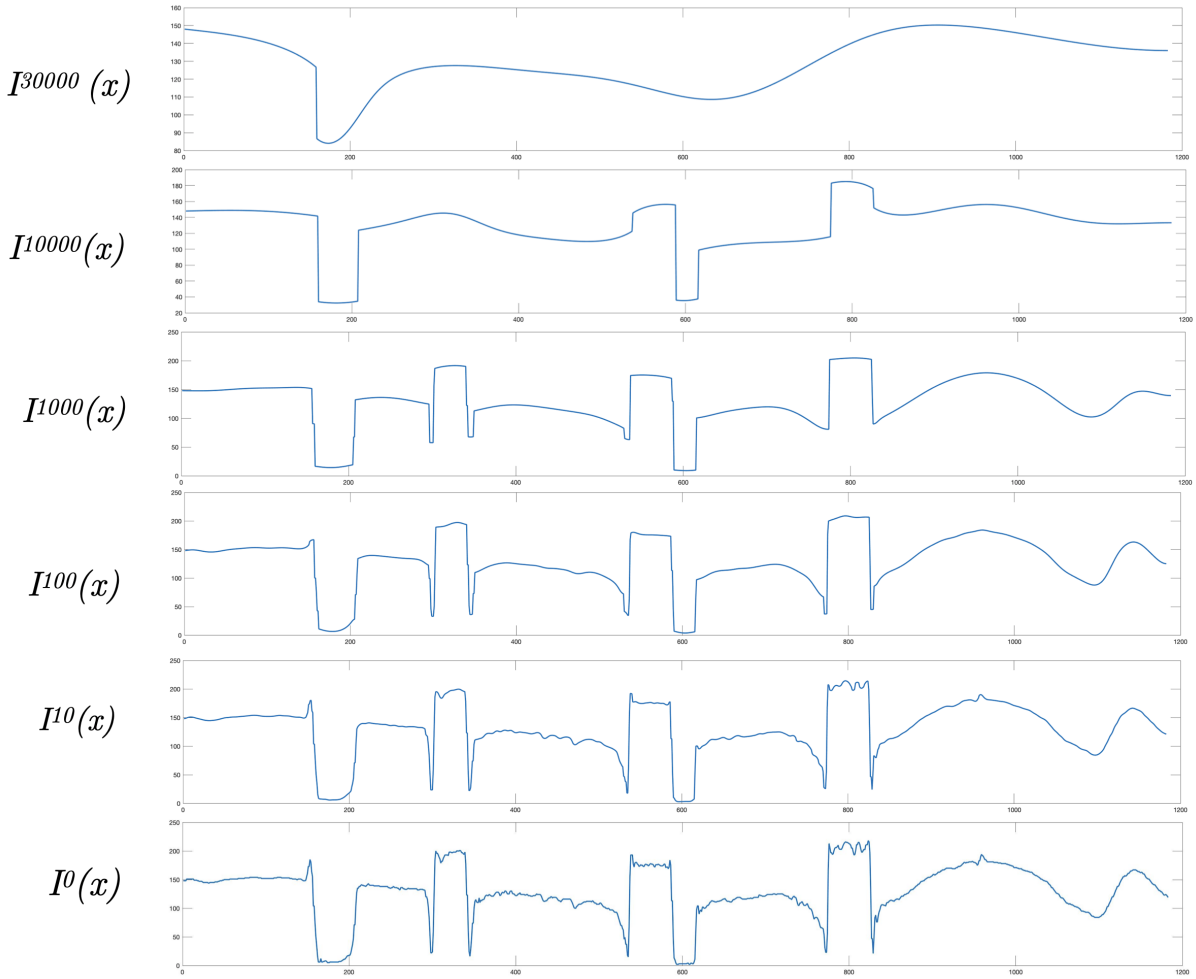

Figura 14: Imágenes obtenidas tras aplicar el filtro de Perona-Malik 0, 10, 100, 1000, 10000 y 30000 veces, con

$$g(|I_x|) = \frac{C}{1 + \left(\frac{|I_x|}{K}\right)^{1+\alpha}}$$

y tomando $C = 1$, $\alpha = 1$ y $K = 4$. Gráficas obtenidas con Matlab 2016b.

Como se puede observar, este método ofrece claros avances con respecto al filtro gaussiano. En las 100 iteraciones, bordes como los de 150, 300 o 350, que en el filtro gaussiano se perdían, aquí se mantienen. Con 1000 iteraciones, ya se ha suavizado prácticamente toda la imagen, sin perder la mayoría de los bordes. Sin embargo, en $t = 10000$ ya se han perdido casi todos los detalles de la imagen, y en $t = 30000$, la imagen ya ha perdido todo su significado.

Por este motivo, trataremos de precisar más este método, con el objetivo de mejorar los resultados:

La idea más directa sería reducir nuestra cota para los bordes $\lambda = \frac{K}{\alpha+1\sqrt{\alpha}}$, a la que hemos dado valor 4 con $\alpha = 1$ y $K = 4$. Sin embargo, $\lambda = 4$ ya es un valor bastante bajo, y reducir el valor de K no parece resolver los problemas.

Por otro lado, modificar el valor de α sí que provoca cambios significativos. Simplemente

tomando $\alpha = 3$ y $K = 4 \cdot \sqrt[4]{3}$, obtendremos resultados mucho mejores.
Esto se debe a que la nueva función g decrece de una manera mucho más brusca, lo que hace que las aproximaciones vistas en el apartado **3.1** sean mucho más precisas. g' tomará valores cercanos a 0 en una región más amplia de $|I_x| = 0$, y tanto g como g' tenderán mucho más rápido a 0 cuando $|I_x|$ crezca.

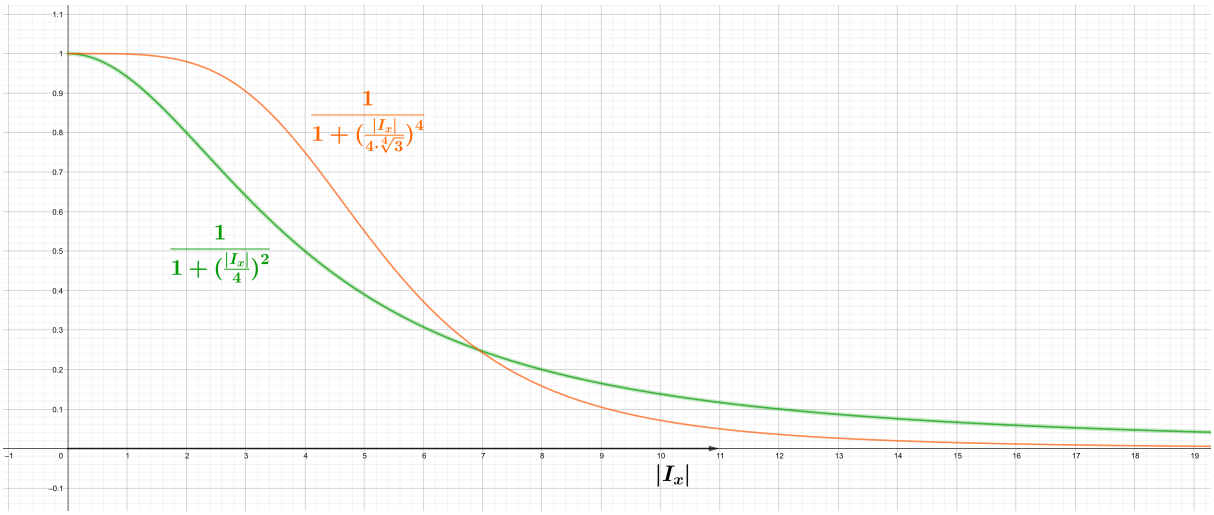

Figura 15: Comparativa de las gráficas de las dos funciones g; una con $C = 1$, $\alpha = 1$ y $K = 4$, y la otra, con $C = 1$, $\alpha = 3$ y $K = 4 \cdot \sqrt[4]{3}$. Gráficas obtenidas con GeoGebra Classic 6.

Realizando este cambio, los resultados claramente mejoran:

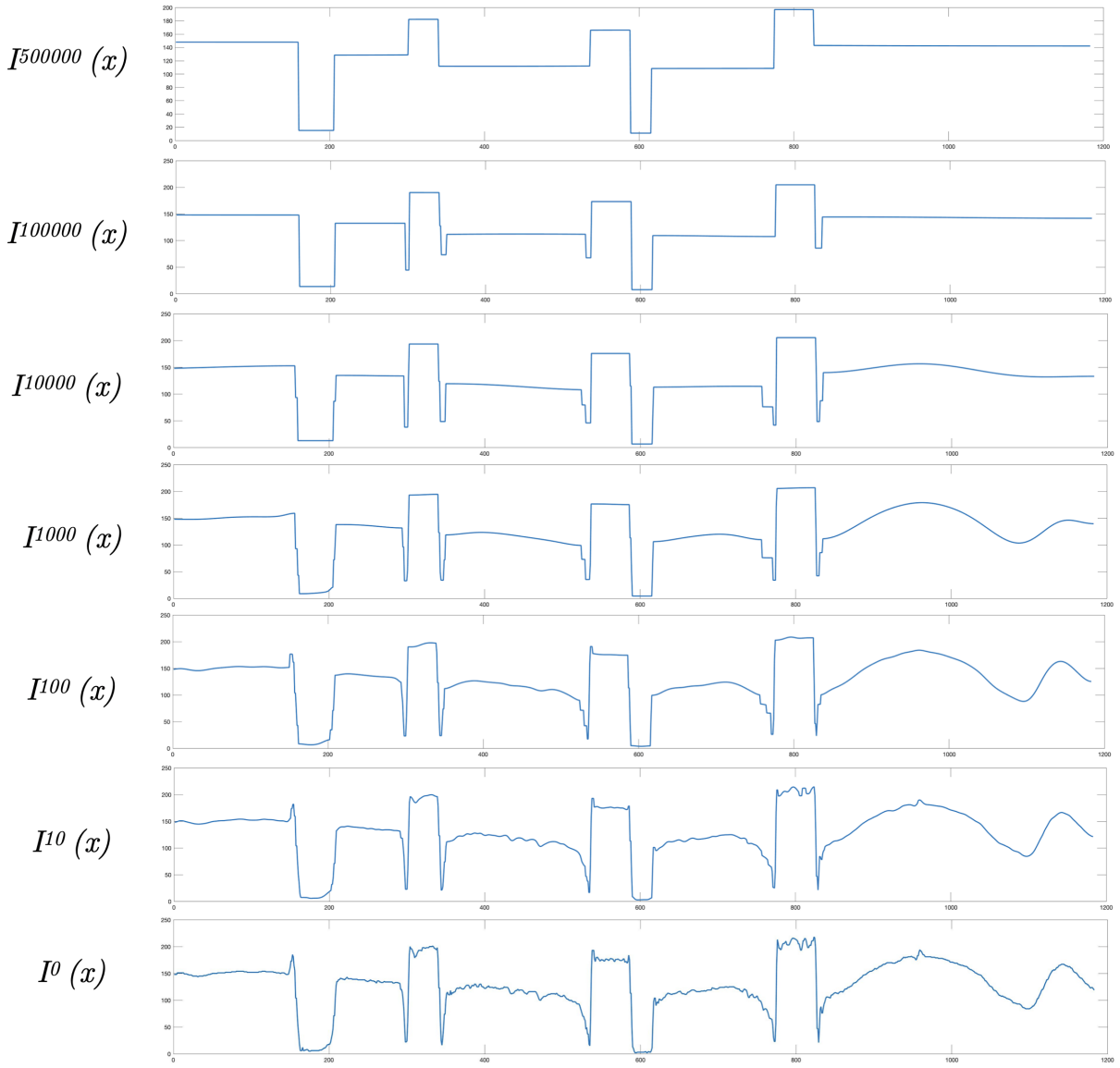

Figura 16: Imágenes obtenidas tras aplicar el filtro de Perona-Malik 0, 10, 100, 1000, 10000, 100000 y 500000 veces, con

$$g(|I_x|) = \frac{C}{1 + \left(\frac{|I_x|}{K}\right)^{1+\alpha}}$$

y tomando $C = 1$, $\alpha = 3$ y $K = 4 \cdot \sqrt[4]{3}$. Gráficas obtenidas con Matlab 2016b.

Finalmente, hemos conseguido un filtro que conserva el significado de la imagen original para valores grandes de t. Además, en los resultados entre las 1000 y las 10000 iteraciones, la mayoría de regiones ya se suavizan por completo, mientras que se conservan prácticamente todos los bordes originales.
Sin embargo, siguen existiendo algunos bordes, como en 150, que desaparecen con valores muy bajos de t, y otros, mucho más marcados, como los de 540 o 780, que se pierden para una cantidad muy alta de iteraciones. Esto se debe a que tomar la función

$$g(|I_x|) = \frac{C}{1 + \left(\frac{|I_x|}{K}\right)^{1+\alpha}}$$

en general, provoca que el filtro conserve mejor los bordes que están más separados entre sí, es decir, los que delimitan regiones más amplias, que los que están muy juntos.

En la práctica, esto origina que no exista ninguna línea que separe una región de otra. En el caso del pez payaso; en la imagen original, cada región de su cuerpo está limitada por una línea negra, pero tras aplicar el filtro para un t muy alto, la mayoría de esas líneas desaparecerán, y solo quedarán las regiones.

Si queremos conservar esas líneas; de nuevo, modificando la función g, se puede conseguir que el filtro se ajuste a nuestras exigencias.

En este caso, tomaremos $g(|I_x|) = e^{-\left(\frac{|I_x|}{K}\right)^2}$, con $K = \sqrt{2} \cdot \lambda$, como ya habíamos visto. Con esta función, conseguimos que el filtro respete mucho más los bordes muy marcados que encierran regiones pequeñas:

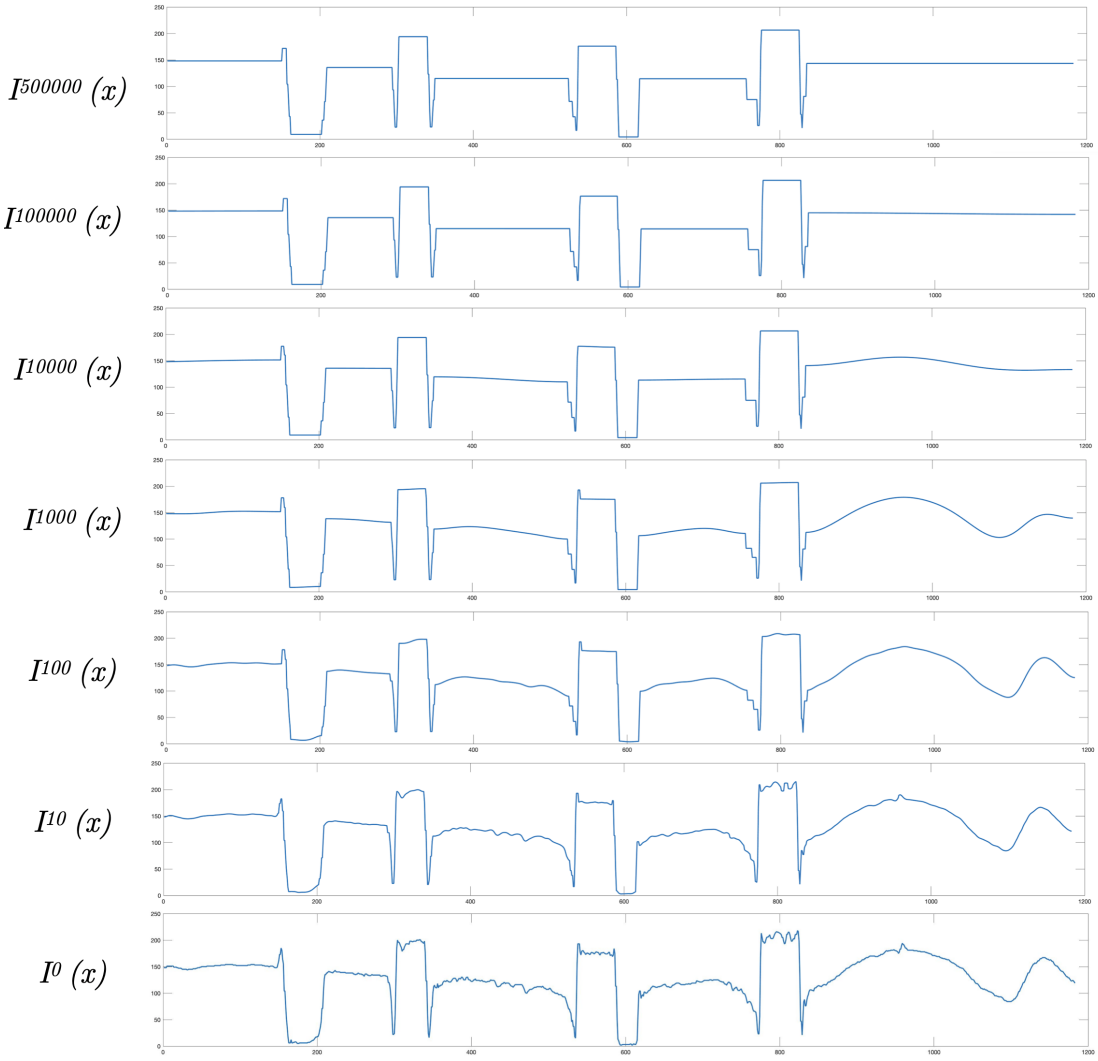

Figura 17: Imágenes obtenidas tras aplicar el filtro de Perona-Malik 0, 10, 100, 1000, 10000, 100000 y 500000 veces, con

$$g(|I_x|) = e^{-\left(\frac{|I_x|}{K}\right)^2}$$

y tomando $K = 4 \cdot \sqrt{2}$. Gráficas obtenidas con Matlab 2016b.

Como se puede apreciar, este filtro también se mantiene estable para cualquier valor t, y es similar al anterior, con la diferencia de que este conserva los bordes más finos, incluso para $t = 500000$.

Cabe destacar que, como se aprecia a simple vista, en cualquier valor de t, y tomando cualquier función g; por las propiedades matemáticas que ya hemos desarrolado sobre la ecuación de Perona-Malik, este filtro marca de forma mucho más clara los bordes. Mientras que en el filtro gaussiano los bordes son más curvados y suaves, con el filtro de Perona-Malik, los bordes se vuelven más agudos y sus gráficas se caracterizan por una gran cantidad de ángulos de 90°.

4. Implementación de filtros en imágenes bidimensionales

En esta sección pasaremos a trabajar con filtros en imágenes en dos dimensiones, los cuales son más complicados, pero claramente más útiles.

Dados $n, m \in \mathbb{N}$. Sea $I : \{0, 1, 2, ..., n\} \times \{0, 1, 2, ..., m\} \longrightarrow \mathbb{R}$ una función bidimensional, y sean $x_0 \in \{0, 1, 2, ..., n\}$ e $y_0 \in \{0, 1, 2, ..., m\}$; se definen:

$$I_{x^+}(x_0, y_0) := \frac{I(x_0 + 1, y_0) - I(x_0, y_0)}{\delta x}$$
$$I_{x^-}(x_0, y_0) := \frac{I(x_0, y_0) - I(x_0 - 1, y_0)}{\delta x}$$
$$I_{y^+}(x_0, y_0) := \frac{I(x_0, y_0 + 1) - I(x_0, y_0)}{\delta y}$$
$$I_{y^-}(x_0, y_0) := \frac{I(x_0, y_0) - I(x_0, y_0 - 1)}{\delta y}$$

Nótese que estas definiciones son análogas a las de la **Sección 3**, con la excepción de que, en un principio, no fijaremos $\delta x = 1$, ni $\delta y = 1$. En este caso, tomaremos $\delta x = \frac{1}{n}$ y $\delta y = \frac{1}{m}$.

Igualmente, a partir de lo estudiado en la sección anterior, las derivadas segundas se construirán como:

$$I_{xx}(x_0, y_0) = (I_{x^+})_{x^-}(x_0, y_0) = \frac{I_{x^+}(x_0, y_0) - I_{x^+}(x_0 - 1, y_0)}{\delta x} =$$
$$\frac{\frac{I(x_0+1,y_0)-I(x_0,y_0)}{\delta x} - \frac{\left(I(x_0,y_0)-I(x_0-1,y_0)\right)}{\delta x}}{\delta x} =$$
$$\frac{I(x_0 + 1, y_0) + I(x_0 - 1, y_0) - 2 \cdot I(x_0, y_0)}{\delta x^2}$$

$$I_{yy}(x_0,y_0) = (I_{y^+})_{y^-}(x_0,y_0) = \frac{I_{y^+}(x_0,y_0) - I_{y^+}(x_0,y_0-1)}{\delta y} = \frac{\frac{I(x_0,y_0+1)-I(x_0,y_0)}{\delta y} - \frac{\big(I(x_0,y_0)-I(x_0,y_0-1)\big)}{\delta y}}{\delta y} = \frac{I(x_0,y_0+1)+I(x_0,y_0-1)-2\cdot I(x_0,y_0)}{\delta y^2}$$

4.1. Implementación del Filtro Gaussiano

Dado que ahora estamos trabajando en dos dimensiones, el Número de Courant-Friedrichs-Lewy pasa a ser $\Lambda := \frac{\delta t}{\delta x^2} + \frac{\delta t}{\delta y^2}$. Como veremos en la **Sección 4.2**, para asegurar la convergencia del método, será necesario que $\Lambda \in (0, \frac{1}{2}]$.

Con todo esto, un posible valor de δt, que asegura la convergencia del método, sería:

$$\delta t = \frac{(min\{\delta x, \delta y\})^2}{4}$$

ya que

$$\Lambda = \frac{\delta t}{\delta x^2} + \frac{\delta t}{\delta y^2} = \frac{\frac{(min\{\delta x,\delta y\})^2}{4}}{\delta x^2} + \frac{\frac{(min\{\delta x,\delta y\})^2}{4}}{\delta y^2} \leq \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

Entonces, la ecuación del calor discretizada será:

$$I^{t+1}(x_0,y_0) = I^t(x_0,y_0) + \frac{\frac{(min\{\delta x,\delta y\})^2}{4}}{\delta x^2} \cdot \big(I(x_0+1,y_0) + I(x_0-1,y_0) - 2\cdot I(x_0,y_0)\big) + \\ + \frac{\frac{(min\{\delta x,\delta y\})^2}{4}}{\delta y^2} \cdot \big(I(x_0,y_0+1) + I(x_0,y_0-1) - 2\cdot I(x_0,y_0)\big)$$

Mediante esta ecuación, se ha construido el filtro gaussiano que se aplica en las imágenes de la **Sección 1.2**. A continuación se expone otra imagen a la que se le aplica el filtro:

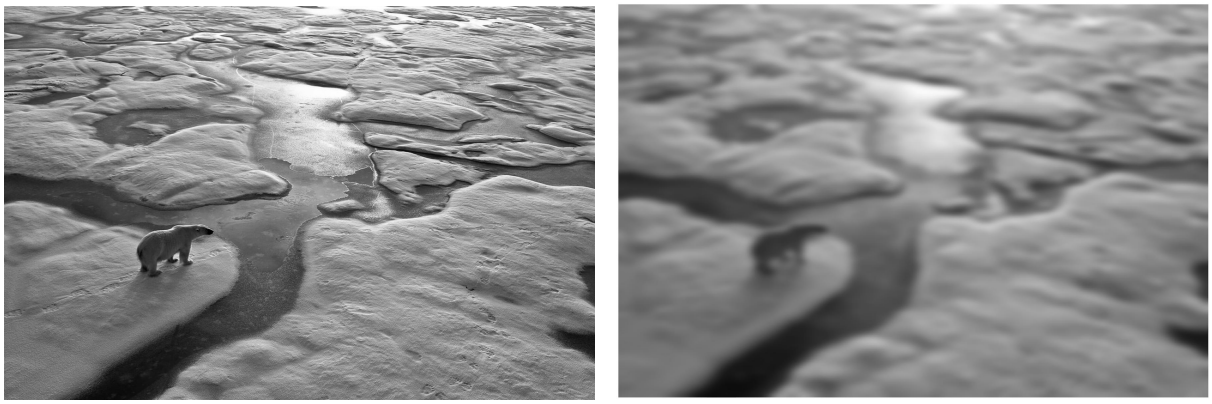

Figura 18: Imagen original de un oso polar en el Ártico, obtenida de [6], y, a la derecha, la imagen tras aplicarle el filtro gaussiano con 200 iteraciones, construida con Matlab 2016b.

Como se puede apreciar, tras aplicar el filtro resulta difícil distinguir las zonas de nieve con las de hielo o agua, sobre todo en la parte superior de la imagen. Es decir, en vez de ensalzar las distintas regiones, lo que se consigue es difuminarlas entre sí, haciendo difícil incluso entender el significado original de la imagen.

4.2. Implementación del Filtro de Perona-Malik

De manera análoga al caso unidimensional, la ecuación (3) se discretizará como:

$$\begin{aligned}
&\frac{I^{t+1}(x_0,y_0)-I^t(x_0,y_0)}{\delta t} = \\
&\left(g\Big(\sqrt{(I^t_{x^+})^2+(I^t_{y^+})^2}\Big)\cdot I^t_{x^+}\right)_{x^-}(x_0,y_0)+\left(g\Big(\sqrt{(I^t_{x^+})^2+(I^t_{y^+})^2}\Big)\cdot I^t_{y^+}\right)_{y^-}(x_0,y_0) = \\
&= \frac{\left(g\Big(\sqrt{(I^t_{x^+})^2+(I^t_{y^+})^2}\Big)\cdot I^t_{x^+}\right)(x_0,y_0)-\left(g\Big(\sqrt{(I^t_{x^+})^2+(I^t_{y^+})^2}\Big)\cdot I^t_{x^+}\right)(x_0-1,y_0)}{\delta x}+ \\
&+\frac{\left(g\Big(\sqrt{(I^t_{x^+})^2+(I^t_{y^+})^2}\Big)\cdot I^t_{y^+}\right)(x_0,y_0)-\left(g\Big(\sqrt{(I^t_{x^+})^2+(I^t_{y^+})^2}\Big)\cdot I^t_{y^+}\right)(x_0,y_0-1)}{\delta y} = \\
&= g\Big(\sqrt{\frac{\big(I^t(x_0+1,y_0)-I^t(x_0,y_0)\big)^2}{\delta x^2}+\frac{\big(I^t(x_0,y_0+1)-I^t(x_0,y_0)\big)^2}{\delta y^2}}\Big)\cdot \\
&\qquad \cdot\frac{\left(\frac{I^t(x_0+1,y_0)-I^t(x_0,y_0)}{\delta x}\right)}{\delta x}- \\
&g\Big(\sqrt{\frac{\big(I^t(x_0,y_0)-I^t(x_0-1,y_0)\big)^2}{\delta x^2}+\frac{\big(I^t(x_0-1,y_0+1)-I^t(x_0-1,y_0)\big)^2}{\delta y^2}}\Big)\cdot \\
&\qquad \cdot\frac{\left(\frac{I^t(x_0,y_0)-I^t(x_0-1,y_0)}{\delta x}\right)}{\delta x}+ \\
&+g\Big(\sqrt{\frac{\big(I^t(x_0+1,y_0)-I^t(x_0,y_0)\big)^2}{\delta x^2}+\frac{\big(I^t(x_0,y_0+1)-I^t(x_0,y_0)\big)^2}{\delta y^2}}\Big)\cdot \\
&\qquad \cdot\frac{\left(\frac{I^t(x_0,y_0+1)-I^t(x_0,y_0)}{\delta y}\right)}{\delta y}- \\
&g\Big(\sqrt{\frac{\big(I^t(x_0+1,y_0-1)-I^t(x_0,y_0-1)\big)^2}{\delta x^2}+\frac{\big(I^t(x_0,y_0)-I^t(x_0,y_0-1)\big)^2}{\delta y^2}}\Big)\cdot \\
&\qquad \cdot\frac{\left(\frac{I^t(x_0,y_0)-I^t(x_0,y_0-1)}{\delta y}\right)}{\delta y}
\end{aligned} \tag{8}$$

Ahora, de manera similar al caso unidimensional, demostraremos un teorema que nos asegura la convergencia del método cuando tomamos un paso temporal δt tal que el

Número de Courant-Friedrichs-Lewy, $\Lambda = \frac{\delta t}{\delta x^2} + \frac{\delta t}{\delta y^2}$ pertenezca a $(0, \frac{1}{2}]$:

Teorema 3 (**Principio del Máximo y del Mínimo Discretos**). *Sean $n, m \in \mathbb{N}$. Dada una secuencia de funciones $I^0, I^1, ..., I^{t_0}$, donde $I^t : \{1, 2, ..., n\} \times \{1, 2, ..., m\} \to \{0, 1, ..., 255\}$ $\forall t \in \{0, 1, ..., t_0\}$, que satisface la ecuación (8) cuando $0 < \frac{\delta t}{\delta x^2} + \frac{\delta t}{\delta y^2} \leq \frac{1}{2}$ y $g : \mathbb{R}^+ \to [0, 1]$; se tiene que el valor máximo y el mínimo de entre todas las funciones lo alcanza I^0, es decir, siendo $I^t_{min} = \min_{(x_0,y_0)\in\{1,2,...,n\}\times\{1,2,...,m\}} \big(I^t(x_0, y_0)\big)$, y $I^t_M = \max_{(x_0,y_0)\in\{1,2,...,n\}\times\{1,2,...,m\}} \big(I^t(x_0, y_0)\big)$, se cumple que:*

$$I^0_{min} \leq I^t(x, y) \leq I^0_M \quad \forall t \in \{0, 1, ..., t_0\}, \forall (x, y) \in \{1, 2, ..., n\} \times \{1, 2, ..., m\}$$

Demostración:

Dado $x_0 \in \{1, 2, ..., n\}$, e $y_0 \in \{1, 2, ..., m\}$; para simplificar la notación, llamaremos

$$g_1 := g\Big(\sqrt{\frac{\big(I^t(x_0+1, y_0) - I^t(x_0, y_0)\big)^2}{\delta x^2} + \frac{\big(I^t(x_0, y_0+1) - I^t(x_0, y_0)\big)^2}{\delta y^2}}\Big)$$

$$g_2 := g\Big(\sqrt{\frac{\big(I^t(x_0, y_0) - I^t(x_0-1, y_0)\big)^2}{\delta x^2} + \frac{\big(I^t(x_0-1, y_0+1) - I^t(x_0-1, y_0)\big)^2}{\delta y^2}}\Big)$$

$$g_3 := g\Big(\sqrt{\frac{\big(I^t(x_0+1, y_0) - I^t(x_0, y_0)\big)^2}{\delta x^2} + \frac{\big(I^t(x_0, y_0+1) - I^t(x_0, y_0)\big)^2}{\delta y^2}}\Big)$$

$$g_4 := g\Big(\sqrt{\frac{\big(I^t(x_0+1, y_0-1) - I^t(x_0, y_0-1)\big)^2}{\delta x^2} + \frac{\big(I^t(x_0, y_0) - I^t(x_0, y_0-1)\big)^2}{\delta y^2}}\Big)$$

A partir de la ecuación (8), tenemos:

$$\begin{aligned} I^{t+1}(x_0, y_0) = I^t(x_0, y_0) + \delta t \cdot \Big(g_1 \cdot \frac{I^t(x_0+1, y_0) - I^t(x_0, y_0)}{\delta x^2} - g_2 \cdot \frac{I^t(x_0, y_0) - I^t(x_0-1, y_0)}{\delta x^2} + \\ + g_3 \cdot \frac{I^t(x_0, y_0+1) - I^t(x_0, y_0)}{\delta y^2} - g_4 \cdot \frac{I^t(x_0, y_0) - I^t(x_0, y_0-1)}{\delta y^2}\Big) = \\ = I^t(x_0, y_0) \cdot \Big(1 - \big(\frac{\delta t}{\delta x^2} \cdot (g_1 + g_2) + \frac{\delta t}{\delta y^2} \cdot (g_3 + g_4)\big)\Big) + \\ + \frac{\delta t}{\delta x^2} \cdot \big(g_1 \cdot I^t(x_0+1, y_0) + g_2 \cdot I^t(x_0-1, y_0)\big) + \frac{\delta t}{\delta y^2} \cdot \big(g_3 \cdot I^t(x_0, y_0+1) + g_4 \cdot I^t(x_0, y_0-1)\big) \end{aligned}$$

Con las hipótesis del enunciado, se cumple que:

$$0 \leq \frac{\delta t}{\delta x^2} \cdot (g_1 + g_2) + \frac{\delta t}{\delta y^2} \cdot (g_3 + g_4) \leq \frac{\delta t}{\delta x^2} \cdot 2 + \frac{\delta t}{\delta y^2} \cdot 2 = 2 \cdot \big(\frac{\delta t}{\delta x^2} + \frac{\delta t}{\delta y^2}\big) \leq 1$$

es decir,

$$1 - \Big(\frac{\delta t}{\delta x^2} \cdot (g_1 + g_2) + \frac{\delta t}{\delta y^2} \cdot (g_3 + g_4)\Big) \geq 0$$

Entonces:

$$I^{t+1}(x_0,y_0) \leq I_M^t \cdot \Big(1 - \big(\frac{\delta t}{\delta x^2}\cdot(g_1+g_2) + \frac{\delta t}{\delta y^2}\cdot(g_3+g_4)\big)\Big) +$$
$$+\frac{\delta t}{\delta x^2}\cdot\big(g_1\cdot I_M^t + g_2\cdot I_M^t\big) + \frac{\delta t}{\delta y^2}\cdot\big(g_3\cdot I_M^t + g_4\cdot I_M^t\big) =$$
$$= I_M^t - I_M^t\cdot\big(\frac{\delta t}{\delta x^2}\cdot(g_1+g_2) + \frac{\delta t}{\delta y^2}\cdot(g_3+g_4)\big) + I_M^t\cdot\big(\frac{\delta t}{\delta x^2}\cdot(g_1+g_2) + \frac{\delta t}{\delta y^2}\cdot(g_3+g_4)\big) =$$
$$= I_M^t$$

En particular,

$$I_M^0 \geq I_M^1 \geq I_M^2 \geq ... \geq I_M^{t_0}$$

es decir

$$I_M^0 \geq I^t(x_0,y_0) \quad \forall t \in \{0,1,...,t_0\},\ \forall (x_0,y_0) \in \{1,2,...,n\}\times\{1,2,...,m\}$$

De manera análoga:

$$I^{t+1}(x_0,y_0) \geq I_{min}^t \cdot \Big(1 - \big(\frac{\delta t}{\delta x^2}\cdot(g_1+g_2) + \frac{\delta t}{\delta y^2}\cdot(g_3+g_4)\big)\Big) +$$
$$+\frac{\delta t}{\delta x^2}\cdot\big(g_1\cdot I_{min}^t + g_2\cdot I_{min}^t\big) + \frac{\delta t}{\delta y^2}\cdot\big(g_3\cdot I_{min}^t + g_4\cdot I_{min}^t\big) =$$
$$= I_{min}^t - I_{min}^t\cdot\big(\frac{\delta t}{\delta x^2}\cdot(g_1+g_2) + \frac{\delta t}{\delta y^2}\cdot(g_3+g_4)\big) + I_{min}^t\cdot\big(\frac{\delta t}{\delta x^2}\cdot(g_1+g_2) + \frac{\delta t}{\delta y^2}\cdot(g_3+g_4)\big) =$$
$$= I_{min}^t$$

y por tanto

$$I_{min}^0 \leq I^t(x_0,y_0) \quad \forall t \in \{0,1,...,t_0\},\ \forall (x_0,y_0) \in \{1,2,...,n\}\times\{1,2,...,m\}$$

□

Finalmente, aplicaremos este filtro a varias imágenes, y probaremos diferentes funciones g para evaluarlas y compararlas.

Al igual que en la **Sección 3.2**, se utilizarán dos familias de funciones g, que en dos dimensiones serán:

- Exponenciales
$$g(||\nabla I||) = e^{-\left(\frac{||\nabla I||}{K\cdot \text{mín}\{\frac{1}{\delta x},\frac{1}{\delta y}\}}\right)^2}$$
- Racionales
$$g(||\nabla I||) = \frac{1}{1+\left(\frac{||\nabla I||}{K\cdot \text{mín}\{\frac{1}{\delta x},\frac{1}{\delta y}\}}\right)^{1+\alpha}}$$

Nótese que en este caso, al parámetro K se le multiplica por mín$\{\frac{1}{\delta x}, \frac{1}{\delta y}\}$. Esto se hace simplemente para ajustar los valores de K y α, y que estos no difieran demasiado de los que se tomaban en el caso unidimensional.

Por otro lado, para cada imagen, la elección de la función g no es aleatoria. Como se vió en la **Sección 3.2**, las funciones exponenciales respetan mucho más que las racionales los bordes de regiones pequeñas. Por ejemplo, en el caso el pez payaso:

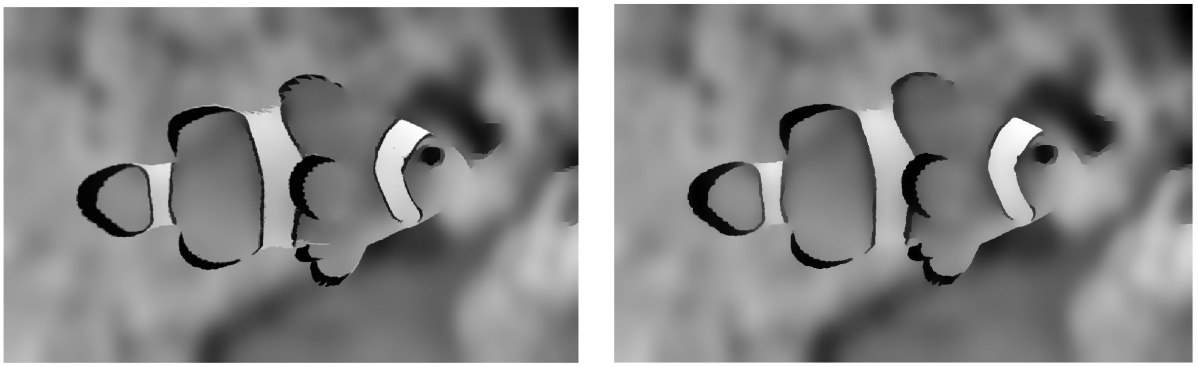

Figura 19: Imagen del pez payaso tras aplicarle el filtro de Perona-Malik (exponencial) con $K = 7$ y 800 iteraciones, a la izquierda. A la derecha, la imagen tras aplicarle el filtro de Perona-Malik (racional) con $K = 4$, $\alpha = 1$ y 800 iteraciones. Imágenes construidas con Matlab 2016b.

Como se puede apreciar, con la función racional, las franjas negras verticales del pez se mezclan con las regiones contiguas, llegando a desaparecer, mientras que con la función exponencial, todas las franjas se conservan. Lo mismo ocurre con los "pelos" blancos que tiene el pez entre sus dos aletas dorsales.

Por tanto, para resaltar todas las regiones, conservando las suficientemente marcadas, pero demasiado pequeñas, se utilizarán funciones exponenciales. Por otro lado, para la imagen de la ragiografía y la de Homer, en las que el objetivo es precisamente eliminar pequeñas imperfecciones relativamente marcadas, se utilizarán funciones racionales.

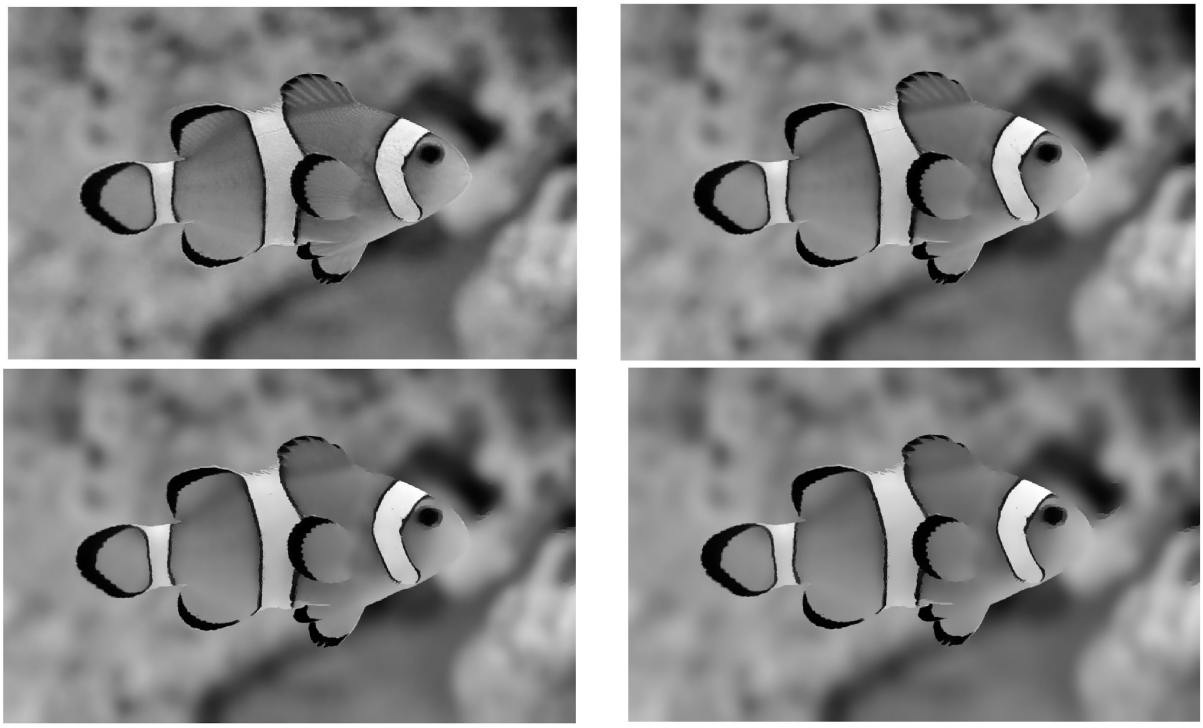

Figura 20: Imagen original del pez payaso arriba a la izquierda. Arriba a la derecha, la imagen tras aplicarle el filtro de Perona-Malik (exponencial) con $K = 7$ y 50 iteraciones. Abajo a la izquierda, tras aplicarle el mismo filtro con 150 iteraciones; y abajo a la derecha, con 300 iteraciones. Imágenes construidas con Matlab 2016b.

En la figura anterior se ve claramente como, a medida que aumenta el número de iteraciones, las escamas del pez se van difuminando, y cada región de su cuerpo (incluidas las franjas negras verticales) se va suavizando y remarcando.

Figura 21: Arriba, la imagen original de un oso polar en el Ártico, obtenida de [6], y, abajo, la imagen tras aplicarle el filtro de Perona-Malik (exponencial) con $K = 6$ y 300 iteraciones, construida con Matlab 2016b.

En este caso, al contrario que el filtro gaussiano, el filtro de Perona-Malik diferencia claramente las regiones de nieve con las de hielo o agua, lo cual provoca incluso que se distingan más claramente que en la imagen original. Igualmente, regiones pequeñas como las huellas del oso se conservan e intensifican, haciendo que destaquen, y facilitando así su localización.

Ahora, para corregir imperfecciones, aplicaremos el filtro de Perona-Malik racional:

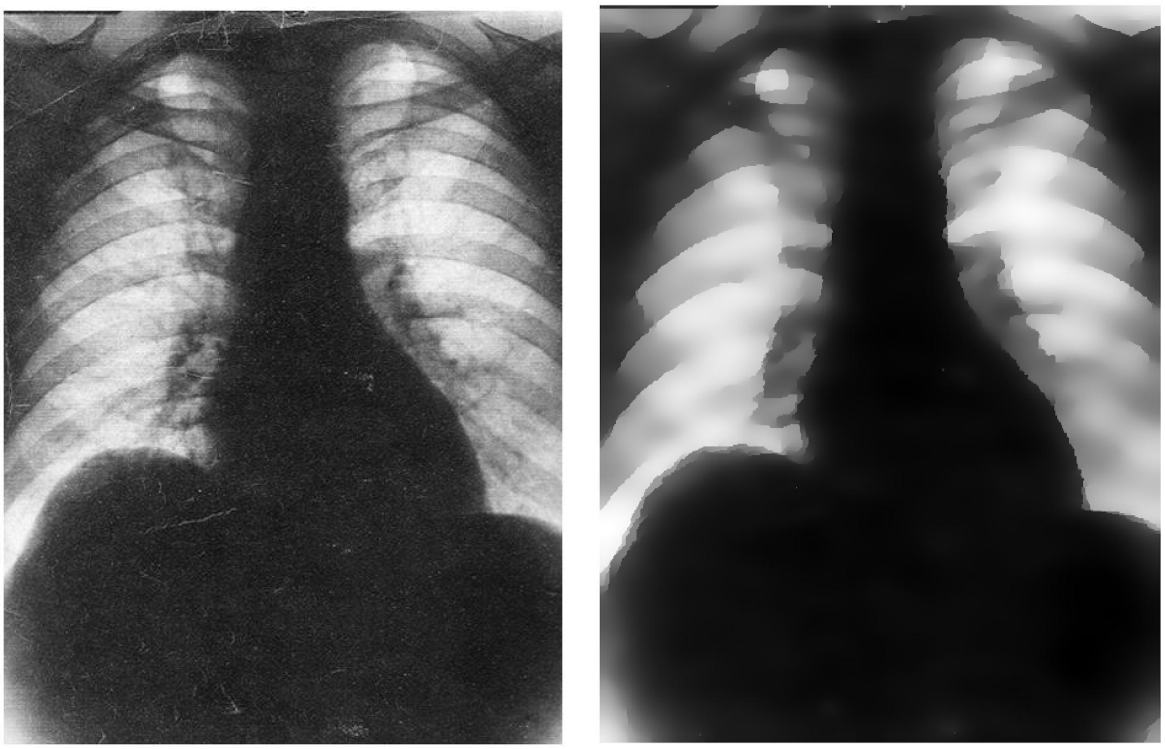

Figura 22: Imagen original del tórax a la izquierda, y, a la derecha, la imagen tras aplicarle el filtro de Perona-Malik (racional) con $K = 4$, $\alpha = 1$ y 200 iteraciones, construida con Matlab 2016b.

Tras aplicar el filtro, hemos eliminado por completo las impurezas de la radiografía y, aún perdiendo cierto nivel detalle, la imagen resultante conserva su significado original.

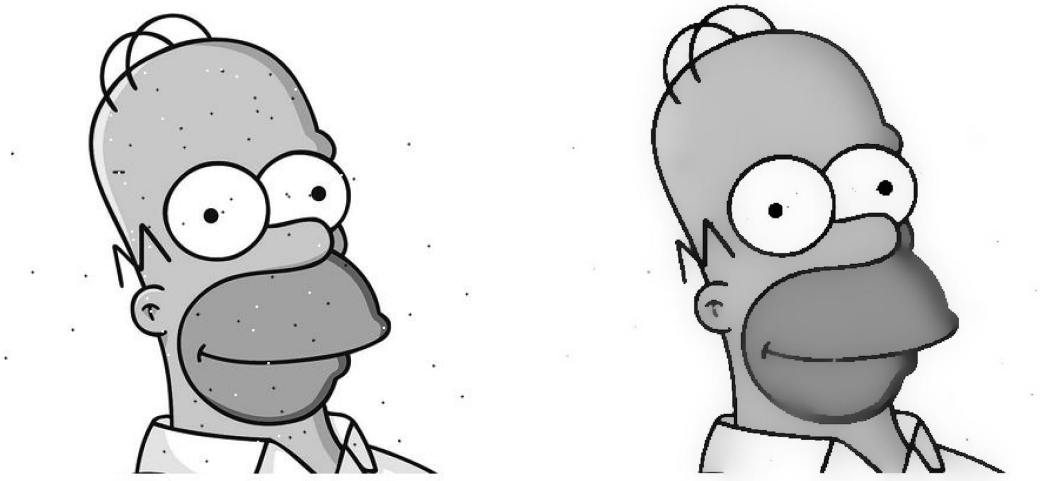

Figura 23: Imagen original de Homer a la izquierda, y, a la derecha, la imagen tras aplicarle el filtro de Perona-Malik (racional) con $K = 8$, $\alpha = 1$ y 550 iteraciones, construida con Matlab 2016b.

En este caso, al tratarse de un dibujo, las distintas regiones están claramente marcadas y diferenciadas, lo que nos permite tomar un valor para K relativamente alto ($K = 8$) sin perder trazos importantes. De esta manera se consiguen eliminar prácticamente todas las manchas sin modificar el resto de la imagen original.

5. Referencias

[1] AllfinWeb. Black fade background the. Disponible en: http://www.allfinweb.com/gallery/black-fade-background-the.html.

[2] Bahamón Cortés, N., et al. *Restauración de imágenes mediante un modelo matemático basado en las técnicas de detección de bordes y propagación de texturas*. PhD thesis, Universidad Nacional de Colombia.

[3] Canny, J. A computational approach to edge detection. In *Readings in computer vision*. Elsevier, 1987, pp. 184–203.

[4] DailyNewa. Quiz: How well do you know homer simpson. Disponible en: http://interactive.nydailynews.com/2016/05/simpsons-quiz/.

[5] Historiadelperonismo.com. Historia del peronismo. Disponible en: https://historiadelperonismo.com/?p=2832.

[6] Iglesias-Suarez, F. El Ártico, un termómetro del cambio climático. Disponible en: https://www.nationalgeographic.com.es/naturaleza/artico-termometro-cambio-climatico_14665.

[7] Miersemann, E. Partial differential equations lecture notes.

[8] MIPAV. Filters (spatial) anisotropic diffusion. Disponible en: https://mipav.cit.nih.gov/pubwiki/index.php/Filters_(Spatial)_Anisotropic_Diffusion.

[9] Peceswiki.com. Pez payaso. Disponible en: https://www.peceswiki.com/pez-payaso.

[10] Perona, P., and Malik, J. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence 12*, 7 (1990), 629–639.

[11] Perona, P., Shiota, T., and Malik, J. Anisotropic diffusion. In *Geometry-driven diffusion in computer vision*. Springer, 1994, pp. 73–92.

[12] Wikipedia. Heat equation. Disponible en: https://en.wikipedia.org/wiki/Heat_equation.

6. Anexo

```
1  %Funcion para aplicar el filtro gaussiano a una imagen
2  %en blanco y negro en dos dimensiones.
3  function [outimage]= gaussiankernel(inimage,iterations)
4
5  %ENTRADA:
6
7      %inimage: imagen inicial, en blanco y negro.
8
9      %iterations: cantidad de veces que se le aplica
10     %            el filtro gaussiano a la imagen.
11
12 %SALIDA:
13
14     %outimage: imagen resultante tras el filtro
15
16 outimage=imread(inimage);
17 [m,n]=size(outimage);
18
19 %Paso la matriz-imagen a una matriz normal para que pueda
20 %tomar valores mayores a 255.
21 outimage=im2double(outimage);
22
23 deltax=1/(m-1);
24 deltay=1/(n-1);
25 deltat=(min(deltax,deltay)^2)/4;
26
27 rowC=[1:m];
28 colC=[1:n];
29 rowN=[1  1:m-1];
30 colE=[1  1:n-1];
31 rowS=[2:m m];
32 colW=[2:n  n];
33
34 sx=(deltat/(deltax^2));
35 sy=(deltat/(deltay^2));
36
37 %Se aplica el filtro gaussiano el numero de veces indicado.
38 for i=1:iterations
39     outimage=outimage+sy*(outimage(rowC,colW)-2*outimage+outimage
       (rowC,colE))+...
40          +sx*(outimage(rowS,colC)-2*outimage+outimage(rowN,colC));
41 end
42
43 imshow(outimage);
```

```
% Funcion para aplicar el filtro gaussiano a una imagen
% en blanco y negro en una dimension.
function []= gaussiankernel1d(vector,iterations)

%ENTRADA:

    % vector: vector que representa una imagen en 1D, en blanco
      y negro.

    % iterations: cantidad de veces que se le aplica
    %             el filtro gaussiano a la imagen.


lambda=0.25;

n=length(vector);

% Paso la matriz-imagen a una matriz normal para que pueda
% tomar valores mayores a 255.
vector=double(vector);


for i=1:iterations
   vector=vector+lambda*(vector([2:n n]) + vector([1 1:n-1]) -2*
     vector);
end

p=plot(vector);
p.LineWidth = 1.85;
```

```
1  %Funcion para aplicar el filtro de Perona-Malik exponencial
2  %a una imagen en blanco y negro en una dimension.
3  function []= anisotropic1d(vector,iterations,K)
4
5  %ENTRADA:
6
7      %vector: vector que representa una imagen en 1D, en blanco
       y negro.
8
9      %iterations: cantidad de veces que se le aplica
10     %            el filtro gaussiano a la imagen.
11
12     %K: parametro de la funcion g de la ecuacion de Perona-
       Malik.
13
14
15 lambda=0.25;
16
17 n=length(vector);
18
19
20 %Paso la matriz-imagen a una matriz normal para que pueda
21 %tomar valores mayores a 255.
22 vector=double(vector);
23
24
25
26 for i=1:iterations
27
28    delta=vector([2:n n])-vector;
29
30    flujo=delta.*exp(-((1/K)*abs(delta)).^2);
31
32    vector=vector+lambda*(flujo-flujo([1  1:n-1]));
33 end
34
35 p=plot(vector);
36 p.LineWidth = 1.85;
```

```
1  %Funcion para aplicar el filtro de Perona-Malik racional
2  %a una imagen en blanco y negro en una dimension.
3  function []= anisotropic1d_frac(vector,iterations,K,alpha)
4
5  %ENTRADA:
6
7      %vector: vector que representa una imagen en 1D, en blanco
           y negro.
8
9      %iterations: cantidad de veces que se le aplica
10     %            el filtro gaussiano a la imagen.
11
12     %K: parametro de la funcion g de la ecuacion de Perona-
           Malik.
13
14     %alpha: parametro de la funcion g de la ecuacion de Perona-
           Malik.
15
16
17 lambda=0.25;
18
19 n=length(vector);
20
21
22 %Paso la matriz-imagen a una matriz normal para que pueda
23 %tomar valores mayores a 255.
24 vector=double(vector);
25
26
27
28 for i=1:iterations
29
30     delta=vector([2:n n])-vector;
31
32     flujo=delta./(1+(abs(delta)/K).^(1+alpha));
33
34     vector=vector+lambda*(flujo-flujo([1  1:n-1]));
35 end
36
37 p=plot(vector);
38 p.LineWidth = 1.85;
```

```
1  %Funcion para aplicar el filtro de Perona-Malik exponencial
2  %a una imagen en blanco y negro en dos dimensiones.
3  function [outimage] = anisoexp(inimage,iterations,K)
4
5  %ENTRADA:
6
7      %inimage: imagen inicial, en blanco y negro.
8
9      %iterations: cantidad de veces que se le aplica
10     %            el filtro gaussiano a la imagen.
11
12     %K: parametro de la funcion g de la ecuacion de Perona-
           Malik.
13
14 %SALIDA:
15
16     %outimage: imagen resultante tras el filtro
17
18
19
20 outimage=imread(inimage);
21 [m,n] = size(outimage);
22
23 %Paso la matriz-imagen a una matriz normal para que pueda
24 %tomar valores mayores a 255.
25 outimage=double(outimage);
26
27 deltax=1/(m-1);
28 deltay=1/(n-1);
29 deltamin=min(deltax,deltay);
30 deltat=(deltamin^2)/4;
31
32 sx=(deltat/(deltax^2));
33 sy=(deltat/(deltay^2));
34
35 for i=1:iterations
36
37     diffY1 = outimage([2:m m],[1:n]) - outimage;
38
39     diffX1 = outimage([1:m],[2:n n]) - outimage;
40
41     diffY2 = [zeros(1,n); diffY1([1:m-1],[1:n])];
42
43     diffX2 = [zeros(m,1) diffX1([1:m],[1:n-1])];
44
45
```

```
46      g1=exp(-((1/((K/deltamin)^2))*(((diffX1/deltax).^2) + ((
            diffY1/deltay).^2))));
47
48      fluxY1 =diffY1.*g1;
49
50      fluxX1 =diffX1.*g1;
51
52      fluxY2= diffY2.*exp(-((1/((K/deltamin)^2))*((([zeros(1,n);
            diffX1([1:m-1],[1:n])]/deltax).^2) + ((diffY2/deltay).^2)
            )));
53
54      fluxX2= diffX2.*exp(-((1/((K/deltamin)^2))*(((diffX2/deltax)
            .^2) + (([zeros(m,1) diffY1([1:m],[1:n-1])]/deltay).^2)))
            );
55
56
57      outimage=outimage + sy*(fluxY1 - fluxY2) +sx*(fluxX1 -
            fluxX2);
58
59  end;
60
61  imshow(outimage,[]);
```

```
1  %Funcion para aplicar el filtro de Perona-Malik racional
2  %a una imagen en blanco y negro en dos dimensiones.
3  function [outimage] = anisofrac(inimage,iterations,K,alpha)
4
5  %ENTRADA:
6
7      %inimage: imagen inicial, en blanco y negro.
8
9      %iterations: cantidad de veces que se le aplica
10     %            el filtro gaussiano a la imagen.
11
12     %K: parametro de la funcion g de la ecuacion de Perona-
           Malik.
13
14     %alpha: parametro de la funcion g de la ecuacion de Perona-
           Malik.
15
16
17 %SALIDA:
18
19     %outimage: imagen resultante tras el filtro
20
21
22 outimage=imread(inimage);
23 [m,n] = size(outimage);
24
25 %Paso la matriz-imagen a una matriz normal para que pueda
26 %tomar valores mayores a 255.
27 outimage=double(outimage);
28
29 deltax=1/(m-1);
30 deltay=1/(n-1);
31 deltamin=min(deltax,deltay);
32 deltat=(deltamin^2)/4;
33
34 sx=(deltat/(deltax^2));
35 sy=(deltat/(deltay^2));
36
37 for i=1:iterations
38
39     diffY1 = outimage([2:m m],[1:n]) - outimage;
40
41     diffX1 = outimage([1:m],[2:n n]) - outimage;
42
43     diffY2 = [zeros(1,n); diffY1([1:m-1],[1:n])];
44
```

```
45      diffX2 = [zeros(m,1) diffX1([1:m],[1:n-1])];
46
47
48      g1=1./(1+((1/((K/deltamin)^(1+alpha)))*((((diffX1/deltax)
            .^2) + ((diffY1/deltay).^2)).^((1+alpha)/2))));
49
50      fluxY1 =diffY1.*g1;
51
52      fluxX1 =diffX1.*g1;
53
54
55      fluxY2= diffY2.*(1./(1+((1/((K/deltamin)^(1+alpha)))*((((
            zeros(1,n); diffX1([1:m-1],[1:n])]/deltax).^2) + ((diffY2
            /deltay).^2)).^((1+alpha)/2)))));
56
57      fluxX2= diffX2.*(1./(1+((1/((K/deltamin)^(1+alpha)))*((((
            diffX2/deltax).^2) + (([zeros(m,1) diffY1([1:m],[1:n-1])
            ]/deltay).^2)).^((1+alpha)/2)))));
58
59
60      outimage=outimage + sy*(fluxY1 - fluxY2) +sx*(fluxX1 -
            fluxX2);
61
62  end;
63
64  imshow(outimage,[]);
```